DPM & CBT 2024 Pre-proceedings

19th International Workshop on Data Privacy Management

8th International Workshop on Cryptocurrencies and Blockchain Technology

September 19, 2024
Bydgoszcz (Poland)

Contents

Preface

This document comprises the pre-proceedings fo the DPM 2024 (19th International Workshop on Data Privacy Management), and CBT 2024 (8th International Workshop on Cryptocurrencies and Blockchain Technology) to be held in September 19th, 2024 in Bydgoszcz (Poland). The workshops are part of ESORICS 2024 (29th European Symposium on Research in Computer Security).

These pre-proceedings include preprints of the papers to be presented in the workshops and a posterior post-proceedings book will be published in Springer LNCS series with the revised papers presented in the workshops. These preprints are intended to be used by the workshop participants during the workshops and we encourage readers to refer the post-proceedings edition for citation.

Ken Barker
Sergi Delgado-Segura
Joaquin Garcia-Alfaro
Guillermo Navarro-Arribas
Cristina Perez-Sola
(DPM 2024, and CBT 2024 PC Chairs)

19th International Workshop on Data Privacy Management

PC Chairs

Ken Barker (University of Calgary)
Joaquin Garcia-Alfaro (Institut Polytechnique de Paris)
Guillermo Navarro-Arribas (Universitat Autònoma de Barcelona)

Program Committee

Esma Aïmeur (University of Montreal)
Abderrahim Ait Wakrime (Mohammed V University)
Jordi Casas-Roma (Universitat Autonoma de Barcelona)
Jordi Castellà-Roca (Universitat Rovira i Virgili)
Depeng Chen (Anhui University)
Mauro Conti (University of Padua)
Mathieu Cunche (University of Lyon / Inria)
Nora Cuppens-Boulahia (Polytechnique Montreal)
Mila Dalla Preda (University of Verona)
Sabrina De Capitani di Vimercati (Universita degli Studi di Milano)
Jose M. De Fuentes (Universidad Carlos III de Madrid)
Josep Domingo-Ferrer (Universitat Rovira i Virgili)
Sebastien Gambs (Université du Québec à Montréal)
Lorena González Manzano (Universidad Carlos III de Madrid)
M. Emre Gursoy (Koç University)
Guy-Vincent Jourdan (University of Ottawa)
Marc Juarez (University of Edinburgh)
Christos Kalloniatis (University of the Aegean)
Bruce Kapron (University of Victoria)
Sokratis Katsikas (Norwegian University of Science and Technology)
Christophe Kiennert (Telecom SudParis)
Romain Laborde (University Paul Sabatier Toulouse III)
Patrick Lacharme (Ensicaen)
Giovanni Livraga (University of Milan)
Brad Malin (Vanderbilt University)
Lukas Malina (Brno University of Technology)
David Megias (Universitat Oberta de Catalunya)
Chris Mitchell (Royal Holloway, University of London)

Gerardo Pelosi (Politecnico di Milano)
Cristina Perez-Sola (Universitat Utonoma de Barcelona)
Kai Rannenberg (Goethe University Frankfurt)
Isabel Praça (GECAD / ISEP)
Ruben Rios (Universidad de Malaga)
Pierangela Samarati (Universita degli Studi di Milano)
Vicenç Torra (Umeå University)
Alexandre Viejo (Universitat Rovira i Virgili)
Isabel Wagner (University of Basel)
Jens Weber (University of Victoria)
Lena Wiese (University of Göttingen)
Nicola Zannone (Eindhoven University of Technology)

Additional Reviewers

Haoying Zhang
Chao Yan
Sergio Martínez
Sascha Loebner
Cristòfol Daudén-Esmel
Luis Del Vasto Terrientes

8th International Workshop on Cryptocurrencies and Blockchain Technology

PC Chairs

Cristina Pérez-Solà (Universitat Autònoma de Barcelona)
Sergi Delgado-Segura (Chaincode Labs)

Program Committee

Lennart Ante - Blockchain Research Lab
Daniel Augot - INRIA Saclay
Alex Biryukov - University of Luxembourg
Mauro Conti - University of Padua
Vanesa Daza - Universitat Pompeu Fabra
Victor Garcia - Universitat Oberta de Catalunya
Hannes Hartenstein - Karlsruhe Institute of Technology
Jordi Herrera-Joancomarti - Universitat Autonoma de Barcelona
Jiasun Li - George Mason University
Shin'ichiro Matsuo - Virginia Tech and Georgetown University
Jose Luis Muñoz-Tapia - Universitat Politecnica de Catalunya
Guillermo Navarro-Arribas - Universitat Autonoma de Barcelona
Dongming Peng - University of Nebraska-Lincoln
Matteo Signorini - Nokia Bell Labs
Hitesh Tewari - Trinity College Dublin
Florian Tschorsch - Technische Universitat Berlin
Eirini Tsiropoulou - University of New Mexico
Dimitrios Vasilopoulos - IMDEA Software Institute

Additional Reviewers

Masaki Hasegawa
Chhagan Lal
Pablo García Fernández
Marc Leinweber
Chhagan Lal
Hieu Nguyen
Aleksei Udovenko
Marta Bellés Muñoz

Part I

DPM 2024: 19th International Workshop on Data Privacy Management

DPM Session 1: Privacy & Machine Learning

Privacy-Preserving Optimal Parameter Selection for Collaborative Clustering

Maryam Ghasemian[1] and Erman Ayday[1]

Case Western Reserve University, Cleveland, OH, USA
{maryam.ghasemian, erman.ayday}@case.edu

Abstract. This study investigates the optimal selection of parameters for collaborative clustering while ensuring data privacy. We focus on key clustering algorithms within a collaborative framework, where multiple data owners combine their data. A semi-trusted server assists in recommending the most suitable clustering algorithm and its parameters. Our findings indicate that the privacy parameter (ϵ) minimally impacts the server's recommendations, but an increase in ϵ raises the risk of membership inference attacks, where sensitive information might be inferred. To mitigate these risks, we implement differential privacy techniques, particularly the Randomized Response mechanism, to add noise and protect data privacy. Our approach demonstrates that high-quality clustering can be achieved while maintaining data confidentiality, as evidenced by metrics such as the Adjusted Rand Index and Silhouette Score. This study contributes to privacy-aware data sharing, optimal algorithm and parameter selection, and effective communication between data owners and the server.

Keywords: Clustering · Privacy · Differential Privacy · Membership Inference Attack · Data Mining · Machine Learning.

1 Introduction

Clustering, a fundamental technique in unsupervised machine learning, involves identifying patterns in unlabeled data. This process includes feature selection, measuring data similarity, and evaluating algorithms [21, 35]. There are several types of clustering algorithms: partitioning based [34, 25], distribution based [19, 22], density based [6, 3, 31], and hierarchical [24, 11]. Our study concentrates on selecting the optimal hyperparameters for key representative clustering algorithms from each category, within a privacy-preserving collaborative framework. Specifically, we explore K-Means (partitioning-based), Hierarchical Clustering (HC, hierarchical), Gaussian Mixture Models (GMM, distribution-based), and DBSCAN (density-based).Choosing the right parameters is crucial as it directly impacts the accuracy and effectiveness of the clustering results, thereby influencing the insights derived from the data while maintaining privacy.

Motivated by the fact that clustering algorithm perform better with larger amount of data and that datasets are typically distributed across different parties, cooperative clustering and collaborative clustering [7] techniques have been popular. In cooperative clustering, each party generates its own clustering results, and a final clustering is performed via a post-processing step once individual processes are completed. In contrast, collaborative clustering aims to leverage

the contributions of multiple parties by exchanging information about local data, current hypothesized clustering, or algorithm parameters to benefit each other's computations. Due to privacy concerns of the parties, privacy-preserving algorithms have been proposed during collaborative clustering, which aim to protect the sensitive information in each parties' local dataset. However, depending on the type of clustering (partitioning-based, distribution-based, density-based, or hierarchical clustering), parties need to decide on some common input parameters. Selection of such parameters significantly effect the accuracy of the clustering algorithm and existing privacy-preserving collaborative clustering techniques assume such parameters are pre-selected. On the other hand, such parameters typically depend on the distribution of the federated dataset of the parties and they should be determined in a privacy-preserving way before the collaborative clustering. In addition, parties also need to decide the type of the clustering algorithm depending on their federated dataset as different types of algorithms perform differently in particular datasets. To fill this gap, we focus on a server-assisted scenario for collaborative clustering, aiming to evaluate server-provided input parameters and clustering algorithms. We experiment with K-Means, Hierarchical Clustering, Gaussian Mixture Models, and DBSCAN using a labeled numeric dataset, assessing results with metrics like Adjusted Rand Index (ARI) and Silhouette Score.

Using a semi-trusted server enhances privacy and helps select optimal clustering algorithms and parameters without burdening data owners with large computational resources. Differential privacy techniques safeguard data throughout the process. Our findings show that this approach effectively maintains data privacy while delivering high-quality clustering, evidenced by ARI and Silhouette Scores. The Randomized Response mechanism efficiently preserves data structure while protecting privacy.

In this work, we make the following contributions to the context of collaborative clustering with hyper parameter recommendation:

1. Privacy-Preserving and Efficient Communication: We introduce a novel privacy-preserving step in the collaborative clustering process, where data owners share parts of their datasets with the server after applying the randomized response (RR) mechanism to add noise to their respective datasets. This step enhances privacy protection by concealing sensitive information while still allowing for meaningful analysis. Additionally, we establish a seamless communication framework between the data owners and the server, ensuring privacy-preserving data sharing. Unlike previous works that primarily rely on pre-selected clustering parameters and then apply encryption techniques in distributed or collaborative clustering, our approach goes beyond by addressing the challenge of parameter selection by determining the optimal clustering algorithm along with the respective hyper-parameters and incorporating the randomized response (RR) mechanism to introduce noise and safeguard sensitive information during data sharing.

2. Optimal Algorithm Selection: The server plays a crucial role in identifying the optimal clustering algorithm and its corresponding hyper-parameters. By employing various methods, the server evaluates different algorithms and provides data owners with recommendations for achieving the best clustering results. This step helps alleviate the burden of algorithm selection and parameter tuning for data owners.

3. Server-Data Owner Interaction: The server communicates chosen algorithms and parameters back to the data owners, ensuring that all parties are aligned with the recommended strategies. This facilitates a coordinated effort that enhances both accuracy and efficiency.

In summary, our study contributes to privacy-aware data sharing, optimal algorithm and hyper-parameter selection, and effective communication between data owners and the server. The results revealed that the amount of noisy data shared and the privacy budget (ϵ) did not significantly affect the server's algorithm and parameter recommendations. However, an increase in the privacy budget was found to elevate the risk of membership inference attacks, suggesting a trade-off between privacy protection and attack vulnerability.

2 Related Work

Our study reviews privacy-preserving approaches in distributed and collaborative clustering, categorized by algorithm types, introduced in Section 1. Existing methods typically use predefined algorithms and hyperparameters, while our contribution dynamically identifies optimal clustering algorithms and hyperparameters to enhance collaborative clustering performance in a privacy-aware manner.

Bi et al.'s PriKPM scheme [5] introduces a privacy-preserving k-prototype clustering method using additive secret sharing to handle mixed data types in cloud environments, addressing privacy concerns. This framework ensures clustering privacy through secure processing by dual servers, validated by experiments demonstrating computational efficiency and accuracy.

Wang et al. [33] propose a privacy-preserving k-means clustering model for IoT, using multi-key fully homomorphic encryption for secure cloud-edge computations. The model optimizes resource use and ensures data privacy through secure communication protocols, demonstrating the feasibility of privacy-sensitive cloud-edge collaborations with minimal overhead.

Further contributions include Jagannathan and Wright's [20], as well as Baby et al.'s [4], protocols for privacy-preserving distributed K-Means clustering, designed for data partitioned arbitrarily. These protocols maintain data confidentiality while following the K-Means algorithm's iterative nature, allowing secure computation of cluster centers and distances without data exposure.

Additionally, Lin et al. [22] present an expectation maximization-based strategy for private clustering across distributed sites, utilizing secure summation to protect horizontally partitioned data. Liu et al. [23] offer privacy-preserving DBSCAN techniques for data distributed in various ways, employing a Multiplication protocol based on additive homomorphic encryption for secure clustering.

Meng et al. [24] introduce privacy-preserving hierarchical clustering algorithms, emphasizing a two-party model that employs homomorphic encryption and garbled circuits. Their approach provides a dendrogram depicting the clustering process, enriched with detailed merge metadata.

These diverse approaches share a common goal of enhancing privacy in collaborative clustering, yet they employ fixed algorithms and parameters. Our study seeks to advance this domain by focusing on adaptive parameter selection to achieve optimal clustering results, reflecting a significant leap toward balancing privacy preservation and analytical utility in collaborative settings. To provide a clearer comparison of the various approaches, Table 1 summarizes the adversary models and system models considered in the related works discussed above.

Table 1: Overview of Adversary and System Models in Related Works

Reference	**Clustering Algorithm**	**System Model**	**Adversary Model**	**Privacy Technique**
Bi et al. [5]	k-Prototype	Cloud-based with dual servers	Semi-honest adversary	Additive Secret Sharing
Wang et al. [33]	k-Means	IoT ecosystem with cloud-edge collaboration	Semi-honest adversary	Multi-Key Fully Homomorphic Encryption
Jagannathan [20], Baby et al. [4]	k-Means	Arbitrarily partitioned data, distributed	Honest-but-curious adversary	Secure Multiparty Computation (SMC)
Lin et al. [22]	Expectation Maximization	Distributed sites with horizontally partitioned data	Honest-but-curious adversary	Secure Summation
Liu et al. [23]	DBSCAN	Distributed with various data partitions	Honest-but-curious adversary	Additive Homomorphic Encryption
Meng et al. [24]	Hierarchical Clustering	Two-party model	Semi-honest adversary	Homomorphic Encryption and Garbled Circuits
Our Work	Multiple (K-Means, HC, GMM, DBSCAN)	Semi-trusted server in collaborative clustering	Semi-honest server, honest-but-curious data owners	Local Differential Privacy, Randomized Response

3 Background

In this section we review some background and definitions of different clustering algorithms and clustering evaluation metrics as well as the local differential privacy.

3.1 Clustering Algorithms

This study explores four clustering algorithms: partitioning-based, distribution-based, density-based, and hierarchical [34, 25, 6, 24, 19, 22, 3, 11]. K-Means, a widely used unsupervised algorithm, partitions data into K non-overlapping clusters by minimizing distances between data points and centroids [34, 25]. Gaussian Mixture Models (GMM) handle clusters with varying sizes and correlations by assuming data is generated from a mixture of Gaussian distributions [19, 22]. DBSCAN identifies clusters of arbitrary shapes based on data density and automatically detects outliers, without needing to predefine the number of clusters, though it is sensitive to its parameters: neighborhood size (Eps) and minimum points ($minpoint$)[6, 3]. Hierarchical clustering creates a tree of clusters without a pre-specified number, using either a bottom-up or top-down approach. It is useful for hierarchical data but is computationally intensive and varies with the linkage criterion used[24, 36, 15, 14, 17].

3.2 Evaluation Metrics for Clustering Algorithms

This section outlines the evaluation metrics used to assess the effectiveness of the proposed privacy-preserving collaborative clustering approach. To measure the performance of our approach, we use the following metrics, each selected for its capability to capture various dimensions of clustering quality and privacy preservation:

Table 2: Table of symbols and notations.

Symbol	Description
D_i	Dataset of each data owner i
ND_i	Noisy data of each data owner i produced as a result of RR
f_{NDi}	Portion of the noisy data, ND_i, shared with server from each data owner i
RR	Randomized Response mechanism
ϵ , eps	epsilon, Privacy Parameter
Eps	Epsilon, Maximum distance between clusters in DBSCAN
k	Number of clusters
ARI	Adjusted Rand Index
CH	Calinski-Harabasz Index
Homo	Homogeneity of the clusters
Comp	Completeness

Adjusted Rand Index (ARI): Measures the similarity between two clusterings, with scores ranging from -1 (independent clusterings) to 1 (perfect agreement). *Higher* ARI values indicate better clustering performance.

Silhouette Coefficient Score: Evaluates cluster cohesion and separation, with scores ranging from -1 to 1. *Higher* values indicate better-defined clusters.

Calinski-Harabasz Index (CH): Measures clustering quality based on the ratio of between-cluster dispersion to within-cluster dispersion. *Higher* CH values indicate better separation between clusters.

Classification Accuracy: We also added classification accuracy to our evaluation framework, a metric that measures the proportion of correct predictions. Although unusual in unsupervised learning tasks like clustering, it helps evaluate how well cluster assignments match predefined labels when known. This metric is key in scenarios with known data classifications, allowing for direct comparison between our privacy-preserving clusters and actual categories.

Table 2 contains a list of symbols and notations used throughout this paper.

3.3 Local Differential Privacy and Randomized Response Mechanism

Local Differential Privacy (LDP) [8, 10] is a more restricted form of traditional differential privacy [9]. Unlike traditional differential privacy, LDP does not rely on a trusted third party and provides a higher level of data protection for users. In LDP, each user modifies their own data before sharing them with a data aggregator. The aggregator only sees the perturbed data, ensuring privacy. An algorithm A satisfies ϵ-local differential privacy (ϵ-LDP) if, for any input values $v1$ and $v2$: $Pr[A(v1) = y] \leq e^{\epsilon} Pr[A(v2) = y]$, This condition holds true for all possible outputs of the algorithm A. The randomized response mechanism is commonly used to achieve $\epsilon - LDP$ [12]. In this mechanism, an individual reports the true value of a single bit of information with probability p and flips the true value with probability $1-p$, following the $(ln\frac{p}{1-p}) - LDP$ property. Although initially defined for binary inputs (e.g., yes/no), the randomized response mechanism can be generalized. To achieve ϵ-LDP, the generalized randomized response mechanism [18] shares the correct value with probability $p = \frac{e^{\epsilon}}{(e^{\epsilon}+m-1)}$ where m is the number of possible states. Each incorrect value is shared with the probability. $q = \frac{1}{(e^{\epsilon}+m-1)}$. A data aggregator collects the perturbed values from individuals and aims to calculate the frequency of values in the population while preserving privacy.

4 System and Threat Models

In this section, we provide an explanation of the system and threat model for privacy-preserving hyper-parameter identification for collaborative clustering.

4.1 System Model

In the proposed system model, the party who aims to collaborate in clustering with other data owners is referred to as the "data owner" (or researcher), while the server represents a third party that assists the data owners in identifying the optimal clustering algorithm and hyper-parameters. Our approach focuses on the preliminary stages before actual clustering occurs in a collaborative environment. Our objective is to identify the optimal algorithm and input parameters for collaborative clustering among multiple data owners who wish to maintain data privacy. As discussed, different types of clustering algorithms perform differently depending on the type and distribution of the datasets, and hence it is crucial to identify the optimal clustering algorithm type beforehand.

Once these optimal conditions are determined, clustering can then be executed using one of the existing algorithms mentioned in Section 3.1. In this context, data owners selectively share differentially private data with a semi-trusted server. This server plays a crucial intermediary role, analyzing the noisy data to recommend the most suitable clustering algorithm and corresponding hyper-parameters for the data received from data owners.

4.2 Threat Model

In this section, we outline the considered threats in our proposed scheme, which involve both the server and the data owners.

Server: In this study, the server is considered semi-honest, indicating it might engage in malicious activities, such as extracting sensitive information from the datasets of the individual parties (data owners), but it honestly follows the protocol execution. The server's role is pivotal, yet poses a risk of privacy violations. Privacy attacks like membership inference [29, 30, 28], deanonymization [29, 26, 27], and attribute inference [29, 13] are concerns. Membership inference attacks aim to determine whether a specific record is in the dataset. Deanonymization attacks link anonymized data to actual identities using external information. Attribute inference attacks deduce sensitive attributes from observed data. In our setting, the most relevant is membership inference, where the server tries to determine if a specific record is part of one of the data owners' datasets, leading to privacy breaches. Our proposed scheme prevents this by sharing only a small, differentially-private portion of the dataset (f_{NDi}), which makes deanonymization more complex and significantly reduces the threat of membership inference.

Data Owners : In our system model, we assume that the parties involved in the collaborative clustering are honest but curious. This means that while they trust each other and do not engage in malicious behavior, they may still be interested in learning about each other's data. This assumption is based on the fact that other literature (such as those in Section 2) has already addressed the challenges posed by malicious or semi-honest data owners in collaborative clustering using privacy-enhancing techniques like homomorphic encryption. In our work, we specifically focus on the task of selecting the optimal algorithm and hyper-parameters for the clustering process. By concentrating on this aspect, we aim to improve the efficiency and effectiveness of collaborative clustering while assuming a cooperative environment among the data owners.

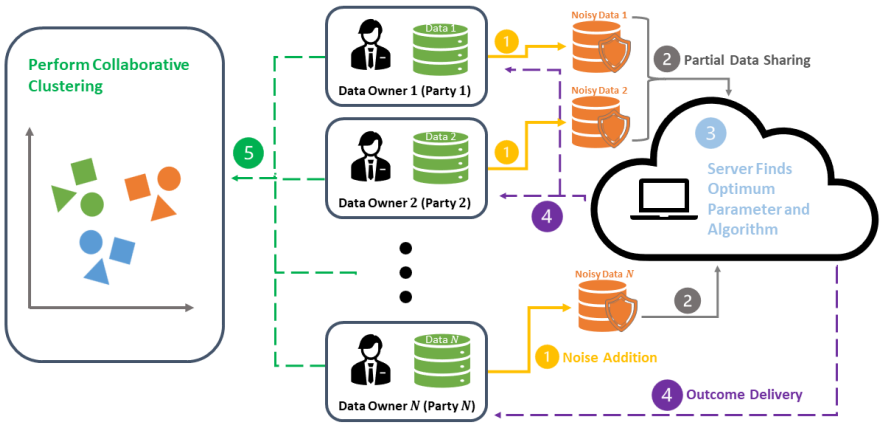


Fig. 1: Comprehensive five-step process, highlighting the interaction between multiple data owners and the server. We show how data are shared, processed for noise addition (to achieve differential privacy), and then utilized in a collaborative clustering algorithm, all while maintaining strict privacy protocols. In step (1), data owners add noise to part of their datasets using randomized response (RR). Data owners send a portion of their noisy data to the server in step (2). In step (3), the server applies various methods to find the optimum algorithm with its corresponding hyper parameter(s), and the server provides its outcome (algorithm and parameter) to the data owners in step (4). Finally, the data owners perform collaborative clustering based on server suggestions in step (5).

5 Proposed Solution and Framework

Our proposed system model and framework, as shown in Figure 1, encompass five fundamental steps:

Step 1-Noise Addition to Datasets: Data owners ($DO_{1\rightarrow N}$) add noise to their datasets (to achieve differential privacy) through randomized response (RR) ($\{D_1, D_2, ..., D_N\} \rightarrow \{ND_1, ND_2, ..., ND_N\}$). In this process, we utilize a generalized version of the RR mechanism as mentioned in Section 3.3, allowing data owners to use perturbed data directly without encoding. The number of possible states for each feature (attribute) can vary according to the specific domain.

Step 2-Data Sharing with the Server: Data owners transmit a portion of their noisy data (f_{NDi}) to the server. During this step, data owners share their perturbed data with the server, enabling it to analyze the data and provide recommendations for the clustering process.

Step 3-Server-Based Algorithm and Parameter Selection : The server selects the best clustering algorithm and its hyperparameters using collaborative clustering, where multiple data owners keep their data private with the Generalized Randomized Response (RR) mechanism. Each owner sends noisy data to a semi-trusted server, which combines the datasets and uses methods like the elbow method and silhouette method [37] to determine optimal parameters for algorithms such as K-Means, hierarchical clustering, Gaussian mixture models. For DBSCAN, it sets the *Eps* value using the k-Nearest Neighbors algorithm and

adjusts the *minpoint* parameter based on data dimensionality, following different recommendations from prior research [14, 32].

One of the challenges the server faces is the absence of ground truth data. To address this, the server uses internal performance evaluation metrics that do not require ground truth, such as the Silhouette Coefficient and the Calinski-Harabasz (CH) index. These metrics objectively measure the effectiveness of different algorithms, guiding the server in its selection process.

Here is the selection mechanism that server adapts to select the optimum clustering algorithm and its corresponding parameters for the data it received from data owners: The input to the selection algorithm includes a combined dataset from all data owners (*data*), a list of candidate clustering algorithms (*algorithms*), and a threshold parameter set to 0.1 (α). The output is the optimal clustering algorithm (*best_ algorithm*) and its corresponding parameters (*best_ parameters*). The procedure begins by initializing *max_ silhouette* to $-\infty$, *best_ algorithm* to *None*, *best_ parameters* to *None*, and *best_ ch_ index* to $-\infty$. It evaluates all algorithms, updating *max_ silhouette* if the Silhouette score is higher. The algorithm then sets a silhouette threshold ($max\_\ silhouette - \alpha$) and selects algorithms within this range with the highest CH index, updating *best_ algorithm*, *best_ parameters*, and *best_ ch_ index* accordingly.

Step 4-Communication of Recommendations: The server communicates the recommended clustering algorithm and its parameters to the data owners, based on the analysis of the shared data.

Step 5-Execution of Collaborative Clustering: Data owners apply the suggested algorithm and hyper-parameters for collaborative clustering. As discussed in Section 2, previous approaches often used encryption for distributed or collaborative clustering. In contrast, this study focuses on selecting the optimal algorithm and hyper-parameters, assuming mutual trust among data owners for clustering on the combined dataset. Further details are provided in Section 4.2.

By following these steps, our framework provides recommendations for the optimal clustering algorithm and its hyper-parameters when data owners wish to perform clustering in a collaborative environment.

6 Evaluation

6.1 Datasets

We use the Obesity dataset [2] (2,111 records, 17 features) and the Extended Iris dataset [1] (1,200 rows, 20 features) which is an enhanced version of the classic Iris dataset [16]. The Obesity dataset assesses obesity levels based on diet and physical condition, while the Extended Iris dataset provides detailed biological and ecological information about the iris flower. These datasets were chosen due to their varying characteristics and complexity, which provide a comprehensive evaluation of our proposed approach across different types of data distributions and clustering challenges.

6.2 Metric Significance and Evaluation Approach

ARI, Silhouette Score, classification accuracy, and Calinski-Harabasz Index (CH) provide a comprehensive performance view. ARI and Silhouette Score assess internal cluster consistency and separation, while classification accuracy offers external validation, and CH highlights cluster distinctness. Together, these metrics enable a thorough assessment of both the clustering effectiveness and the impact of privacy-preserving techniques on data utility. In our evaluation, we analyze

Table 3: Comparison of Silhouette and Elbow Methods for Predicting the Optimal Number of Clusters (k): It highlights the superior performance of the Elbow method in predicting the optimal cluster count, leading to its selection for further analysis in this study.

ϵ	**Baseline K**	**Silhouette K**	**Elbow K**
0.0010	7	2	8
0.1000	7	2	8
1.00	7	2	8
5.00	7	2	7
10.00	7	2	7

these metrics under varying conditions of data perturbation and privacy budget settings to explore the trade-offs between clustering quality and privacy preservation. The goal is to achieve optimal hyper-parameter selection that balances these aspects effectively, demonstrating the practical utility of our approach in collaborative clustering scenarios.

6.3 Evaluation Results

The datasets were pre-processed by converting categorical variables to numerical values for analysis. To determine the optimal number of clusters, we applied the elbow and silhouette methods, as detailed in Section 5. Our experiments, particularly under varying privacy budgets (ϵ), aimed to identify the most effective method for our data. The results for *dataset 1* are shown in Table 3.

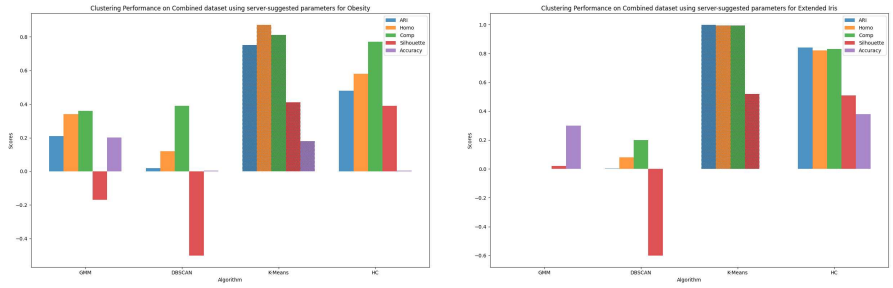
Given that *dataset 1* has 7 clusters and *dataset 2* has 3, our analysis shows that the elbow method outperforms the silhouette method in determining the optimal cluster count. Consequently, we use the elbow method for a more detailed analysis, aiding in the selection of the optimal k for clustering algorithms like K-Means, hierarchical clustering, and Gaussian mixture models.

Optimum Input Parameter Selection Results on Noisy Datasets: The experimental findings of this study are illustrated in Table 4 and Figure 2. Table 4 offers a glimpse into the server's input parameter recommendations, based on the analysis of 10% of the noisy data shared by the data owners, with a noise parameter (ϵ) set at 0.1. Figure 2, on the other hand, showcases the clustering outcomes derived from applying these server recommendations to the combined dataset. Notably, the results from this application highlight the superiority of the K-Means clustering algorithm for the combined dataset, a finding that resonates with the server's initial suggestion regarding the most suitable algorithm and hyper-parameter configuration. These findings and recommendations by the server are not merely data points, but they serve as critical guidance for the data owners. They enable the owners to align their clustering strategies with the server's insights, which are rooted in a meticulous analysis of optimal input parameters. This alignment is key to enhancing the effectiveness and accuracy of the clustering process in a collaborative, privacy-preserving data environment.

Effect of Privacy Parameter ϵ: We have examined the influence of different levels of ϵ, which perturb the data through the Randomized Response (RR) mechanism, on the server's ability to suggest input parameters for clustering algorithms. In this experiment, the server receives the same amount of data while varying the value of ϵ, and its suggestions are evaluated on the joint dataset

Table 4: Server Suggestions for Clustering Input Parameters: Recommendations for various clustering algorithms based on 10% shared noisy data ($\epsilon = 0.1$).

Dataset	Algorithm	Data shared to Server	ϵ	K or Eps	Silhouette	CH
Dataset #1	GMM	10%	0.1	k = 8	0.34	301.30
	DBSCAN	10%	0.1	k = 10, Eps = 1	-	-
	K-Means	**10%**	**0.1**	**k = 8**	**0.36**	**318.13**
	HC	10%	0.1	k = 8	0.31	237.61
Dataset #2	GMM	10%	0.1	k = 3	0.23	46.88
	DBSCAN	10%	0.1	k = 6, Eps = 7	-	-
	K-Means	**10%**	**0.1**	**k = 3**	**0.36**	**61.92**
	HC	10%	0.1	k = 3	0.37	51.57



(a) **(b)**

Fig. 2: Visual Representation of Clustering Algorithm Performance Across Combined Datasets. This figure illustrates the performance metrics from Table 4 for various clustering algorithms—GMM, DBSCAN, K-Means, and Hierarchical Clustering (HC)—evaluated under conditions of 10% data sharing and a privacy parameter of $\epsilon = 0.1$. Performance metrics including Adjusted Rand Index (ARI), Homogeneity (Homo), Completeness (Comp), Silhouette Score, Calinski-Harabasz Index (CH), and Accuracy are plotted. Algorithms recommended by the server are highlighted with dots, showcasing their superior performance in comparison to others in each dataset scenario.

without any noise. Experimental results, as shown in Tables 5 and 6, reveal a notable consistency in the server's recommendations.

Regardless of the ϵ value, the server consistently proposes around 7 clusters for the first dataset (Obesity dataset) and approximately 3 clusters for the second dataset (Extended Iris dataset). This consistency closely aligns with the established ground truth, indicating a marginal effect of the privacy parameter ϵ on the server's cluster count recommendations. However, it is important to note that the actual quality of the clusters formed is subject to the specific clustering algorithm employed. For instance, in the first dataset (Obesity dataset), clustering algorithms demonstrate varied effectiveness influenced by different privacy budgets (ϵ), shown in Table 5. K-Means excel, achieving high ARI values, reaching up to 1.0 when less noise introduced to data (higher ϵ), but maintain low classification accuracy across all settings, indicating well-defined clusters that do not match predefined labels. Silhouette scores also improve with increased ϵ, suggesting clearer cluster definition. Hierarchical Clustering (HC) shows moderate and stable ARI values around 0.48 but face declines in accuracy under extreme privacy settings, hinting at potential misalignments with actual labels. Gaussian Mixture Models (GMM) record lower ARI and negative silhouette scores,

(a) **(b)** **(c)**

Fig. 3: **(a)**:Contrast in dataset #1 with Overlapping Clusters (ϵ = 1, 5, 10): This part displays the differences between original ('O') and noise-modified ('X') data in closely positioned clusters, colored blue and red. **(b)** : Comparison in dataset #1 with Clear Cluster Gaps (ϵ = 1, 5, 10): Here, the focus is on the impact of the Randomized Response (RR) method on data (original 'O', noisy 'X') in maintaining cluster gaps despite noise variations, balancing privacy with data structure integrity. **(c)**: Original vs. Noisy Data in dataset #2 (ϵ = 1, 5, 10): This section compares original ('O') and noise-affected ('X') data at different privacy levels, using blue, red, and green to show cluster separation effectiveness via the RR mechanism. Note: Plots can be zoomed in for clearer visualization.

suggesting less effective clustering and poor separation, with fluctuating accuracy that sometimes aligned with class labels under minimal privacy constraints. DBSCAN consistently performs poorly with very low ARI, negative silhouette scores, and minimal accuracy, indicating its unsuitability for this dataset due to its sensitivity to specific parameter settings and data density. In the second dataset (Extended Iris dataset), the performance of clustering algorithms vary significantly under different privacy settings as shown in Table 6. K-Means showcases excellent clustering with ARI values of 0.997 at low and high ϵ levels, though it drops at $\epsilon = 1$, reflecting its sensitivity to privacy settings, despite maintaining high silhouette scores for good cluster separation. However, its consistently low accuracy indicates a misalignment between the clusters and actual class labels. Hierarchical Clustering (HC) remains stable across all metrics and ϵ settings, achieving moderate to high ARI and silhouette scores, and comparatively better accuracy at 0.38, suggesting it aligns more closely with true labels. Gaussian Mixture Models (GMM) exhibit poor performance with negative ARIs and low silhouette scores, with only moderate accuracy, underscoring its challenges in this dataset under privacy constraints. DBSCAN performs poorly, with extremely low ARI, negative silhouette scores, and zero accuracy across all ϵ settings, confirming its unsuitability for the dataset. Overall, the K-Means algorithm excels over others when the server's recommendation was $k = 3$, according to various evaluation metrics. Furthermore, the server's recommendations do not

significantly deviate from the original data in both datasets. To understand the behavior of data points in dataset #1, we conducted an analysis by selecting two clusters from the original dataset and applying the RR mechanism with varying ϵ values. This investigation revealed that the RR mechanism effectively maintains the separation between clusters when present.

A comparison of Figures 3a and 3b illustrates that the distinction between two randomly selected clusters is retained even when the data is subjected to different ϵ values. This finding is significant as it demonstrates that despite lower ϵ values possibly leading to a more sparse appearance of the data, the server is still capable of accurately identifying two distinct clusters. This is because the RR mechanism ensures that data points are redistributed within a range akin to their original positions. Furthermore, an analysis of Table 4 shows that the server's recommendations for second dataset closely mirror the original data. An exploration involving a comparison of the original and RR-perturbed data points across different ϵ values, as demonstrated in Figure 3c, indicates that in two out of the three clusters in dataset #2, data points overlap without a clear gap, while the third cluster's points are notably distanced from the others. This observation reinforces the notion that the RR mechanism is capable of preserving existing gaps between clusters for various ϵ values.

These results underscore the RR mechanism's proficiency in safeguarding the intrinsic structure of the data while incorporating elements of privacy protection. By effectively maintaining the relative distances between data points, the server is enabled to provide precise recommendations for the number of clusters, despite the noise caused by different ϵ values. This highlights the RR mechanism's balance in protecting data privacy while ensuring the accuracy of clustering algorithm suggestions in a privacy-conscious data analysis setting.

Table 5: Differential Impact of Privacy Levels on Clustering Algorithms in the dataset #1. This table explores the performance variations (measured through ARI, Silhouette, and Accuracy) of four distinct clustering algorithms (K-Means, HC, GMM, DBSCAN) at different privacy budget levels ($\epsilon = 0.1, 1, 5$) with a consistent data sharing percentage (10%).

Algorithm	**Shared**	ϵ	**K**	**ARI**	**Silhouette**	**Accuracy**
K-Means	10%	0.1	k = 8	0.75	0.41	0.18
K-Means	10%	1	k = 8	0.75	0.41	0.18
K-Means	10%	5	k = 7	1	0.44	0.15
HC	10%	0.1	k = 8	0.481	0.39	0.005
HC	10%	1	k = 7	0.482	0.41	0.17
HC	10%	5	k = 8	0.482	0.41	0.005
GMM	10%	0.1	k = 6	0.185	-0.0143	0.201
GMM	10%	1	k = 8	0.2069	-0.072	0.05
GMM	10%	5	k = 6	0.2008	-0.007	0.14
DBSCAN	10%	0.1	k = 10	0.017	-0.504	0.005
DBSCAN	10%	1	k = 10	0.017	-0.504	0.005
DBSCAN	10%	5	k = 10	0.017	-0.504	0.005

Table 6: Influence of Privacy Settings on Clustering Recommendations in the dataset #2. This table details how varying privacy budgets (ϵ = 0.1, 1, 5) affect the recommendations for clustering parameters and subsequent algorithm performance (ARI, Silhouette, and Accuracy) for multiple clustering algorithms (K-Means, HC, GMM, DBSCAN), all with a consistent 10% data sharing arrangement.

Algorithm	Shared	ϵ	K	ARI	Silhouette	Accuracy
K-Means	10%	0.1	k = 3	0.997	0.52	0
K-Means	10%	1	k = 2	0.44	0.57	0.18
K-Means	10%	5	k = 3	0.997	0.52	0
HC	10%	0.1	k = 3	0.84	0.51	0.38
HC	10%	1	k = 3	0.84	0.51	0.38
HC	10%	5	k = 3	0.84	0.51	0.38
GMM	10%	0.1	k = 3	-0.0003	0.021	0.3
GMM	10%	1	k = 2	-0.0004	0.051	0.34
GMM	10%	5	k = 3	-0.0003	0.021	0.3
DBSCAN	10%	0.1	k = 6	0.003	-0.6	0
DBSCAN	10%	1	k = 6	0.003	-0.6	0
DBSCAN	10%	5	k = 6	0.003	-0.6	0

Impact of Shared Data Volume on Server Suggestions: In exploring the influence of shared data volume on clustering algorithm suggestions for both datasets 1 and 2, the results consistently indicate that varying the proportion of data shared with the server does not significantly impact the server's recommendations for clustering input parameters. To investigate this, we conduct experiments where varying amounts of data are shared with the server while keeping the privacy parameter (ϵ) unchanged. This observation is consistent across both datasets and all tested algorithms, as shown in Tables 7 and 8.

For the first dataset, the K-Means algorithm maintains the same ARI, Silhouette, and Accuracy metrics across different data sharing proportions, suggesting that its performance remains stable despite changes in the volume of data shared. Similarly, Hierarchical Clustering (HC), Gaussian Mixture Models (GMM), and DBSCAN show consistent performance metrics across different data sharing amounts, further supporting the notion that the quality of clustering recommendations does not deteriorate with reduced data sharing. In the

Table 7: Impact of Data Sharing Proportions on Clustering Algorithms' Performance in the dataset #1. This table evaluates how different proportions of data shared with the server (10%, 30%, 50%) influence the clustering outcomes (ARI, Silhouette, and Accuracy) for various algorithms (K-Means, HC, GMM, DBSCAN) at a fixed privacy parameter ($\epsilon = 0.1$).

Algorithm	Shared	ϵ	K	ARI	Silhouette	Accuracy
K-Means	10%	0.1	k = 8	0.75	0.41	0.18
K-Means	30%	0.1	k = 8	0.75	0.41	0.18
K-Means	50%	0.1	k = 8	0.75	0.41	0.18
HC	10%	0.1	k = 8	0.481	0.39	0.005
HC	30%	0.1	k = 8	0.481	0.39	0.005
HC	50%	0.1	k = 8	0.0.481	0.39	0.005
GMM	10%	0.1	k = 6	0.185	-0.143	0.201
GMM	30%	0.1	k = 8	0.175	-0.111	0.18
GMM	50%	0.1	k = 5	0.169	-0.001	0.23
DBSCAN	10%	0.1	k = 10	0.017	-0.504	0.005
DBSCAN	30%	0.1	k = 10	0.017	-0.504	0.005
DBSCAN	50%	0.1	k = 10	0.017	-0.504	0.005

Table 8: Analysis of Server Recommendations for Clustering Parameters Based on Data Sharing Amounts in the second Dataset. This table examines the influence of varying amounts of data shared (10%, 30%, 50%) on server-suggested clustering parameter (k) and their resulting ARI, Silhouette, and Accuracy metrics at a constant privacy parameter ($\epsilon = 0.1$).

Algorithm	**Shared**	ϵ	**K**	**ARI**	**Silhouette**	**Accuracy**
K-Means	10%	0.1	k = 3	0.997	0.52	0
K-Means	30%	0.1	k = 2	0.44	0.57	0.18
K-Means	50%	0.1	k = 2	0.44	0.57	0.18
HC	10%	0.1	k = 3	0.84	0.51	0.38
HC	30%	0.1	k = 2	0.55	0.52	0.66
HC	50%	0.1	k = 2	0.55	0.52	0.66
GMM	10%	0.1	k = 3	-0.0003	0.021	0.3
GMM	30%	0.1	k = 2	-0.0004	0.051	0.32
GMM	50%	0.1	k = 2	-0.0004	0.051	0.32
DBSCAN	10%	0.1	k = 6	0.003	-0.6	0
DBSCAN	30%	0.1	k = 6	0.003	-0.6	0
DBSCAN	50%	0.1	k = 6	0.003	-0.6	0

second dataset, similar patterns emerge. For instance, the K-Means algorithm and HC adjust their suggested number of clusters slightly depending on the data share, but the overall performance metrics such as ARI and Silhouette remain relatively stable. This trend continues with GMM and DBSCAN, which also show little variation in performance across different data sharing proportions.

These findings suggest that the server is capable of providing robust and reliable recommendations for clustering parameters regardless of the amount of data shared, enabling effective clustering outcomes even when data owners choose to share minimal data. This is particularly advantageous in scenarios where data privacy is a concern, as it allows data owners to restrict the amount of shared data without compromising the effectiveness of the clustering process. Overall, the server's ability to consistently suggest appropriate clustering parameters across varying data proportions demonstrates its effectiveness and reliability in guiding the clustering process under different data availability conditions.

7 Privacy Analysis: Membership Inference Attack

Membership inference attacks (MIA) are techniques used to determine whether specific individual data was included in a dataset. These attacks pose significant privacy risks, especially when datasets contain sensitive information. Our goal is to minimize these risks for individuals whose data is part of a dataset shared with others. To enhance data privacy, only a portion of the dataset, even in its noisy format, is shared with the server. It has been observed that the likelihood of successful membership inference attacks is inversely related to the amount of noise added to the dataset. We divide the data into two groups to assess the impact of these attacks:

Case Group: This group contains data from specific number of individuals (150 for first dataset and 100 for second dataset) and represents the subset of the dataset that is shared with server, thus exposed to potential membership inference attacks.

Control Group: This group includes data that remains entirely internal and is not shared with the server. It serves as a benchmark to gauge the risk of data exposure. We address membership inference attacks by computing a threshold that determines whether an individual's data is likely part of the training dataset

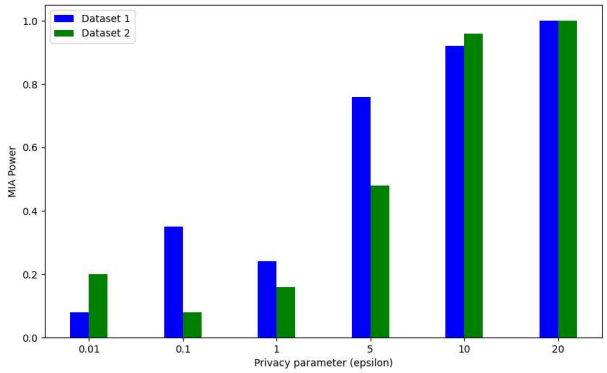


Fig. 4: Analysis of Membership Inference Attack Risks: This figure illustrates the increasing likelihood of data identification in two datasets as privacy parameters (ϵ) increase. The blue bars represent dataset #1, and the green bars represent dataset #2, highlighting the direct correlation between reduced noise levels and heightened data vulnerability.

based on similarity between shared and unshared data. Similarity exceeding this threshold indicates a risk of data exposure through membership inference.

Our detailed analysis is visually represented in Figure 4, demonstrating the impact of the privacy parameter (ϵ), with increased ϵ values reducing the noise and thereby increasing the risk of data identification.

Our findings show that as ϵ increases, the risk of membership inference attacks rises, indicating less noise leads to higher identification likelihood. Therefore, limiting shared data and augmenting it with noise is essential to reduce these risks, necessitating a strategic approach to balance data utility and privacy in collaborative clustering.

8 Conclusion

This study aims to identify optimal input parameters for four clustering algorithms to facilitate collaborative clustering among multiple data owners. Introducing a semi-trusted third party improves clustering reliability and accuracy by recommending optimal algorithms and parameters. Results show that neither the amount of perturbed data shared nor the privacy budget (ϵ) significantly impacts the server's recommendations.

Furthermore, this study conducts an analysis of membership inference attacks to evaluate the vulnerability of the system. As the privacy budget (ϵ) increases, the power of membership inference attacks also increases. This indicates that higher levels of privacy budget compromise the effectiveness of privacy protection, making it easier for attackers to infer whether an individual's data is part of the shared dataset.

These findings emphasize the need for careful consideration of privacy-preserving mechanisms and the importance of maintaining an appropriate balance between privacy protection and utility. While the server's suggestions for input parameters remain consistent regardless of the amount of perturbed data or the privacy budget, the potential risks associated with membership inference attacks

highlight the need to adopt appropriate safeguards and mitigation strategies. Protecting the privacy of individuals and ensuring the security of collaborative clustering processes should be key priorities in future research and system design.

Acknowledgement

The work was partly supported by the National Library of Medicine of the National Institutes of Health under Award Number R01LM013429, the National Science Foundation (NSF) under grant numbers 2141622, 2050410, 2200255, and OAC-2112606, and Cisco Research.

References

1. Extended iris dataset. https://www.kaggle.com/datasets/samybaladram/iris-dataset-extended
2. Estimation of obesity levels based on eating habits and physical condition. UCI Machine Learning Repository (2019), https://doi.org/10.24432/C5H31Z
3. Anikin, I.V., Gazimov, R.M.: Privacy preserving dbscan clustering algorithm for vertically partitioned data in distributed systems. In: Proc. Int. Siberian Conf. Control Commun. (SIBCON) (2017)
4. Baby, V., Chandra, N.S.: Distributed threshold k-means clustering for privacy-preserving data mining. In: Proc. 6th Int. Conf. Adv. Comput. Commun. (ICACCI) (2016)
5. Bi, R., Guo, D., Zhang, Y., Huang, R., Lin, L., Xiong, J.: Outsourced and privacy-preserving collaborative k-prototype clustering for mixed data via additive secret sharing. IEEE Internet Things J. **10**(18), 15810–15821 (2023), https://doi.org/10.1109/JIOT.2023.3266028
6. Bozdemir, B., Canard, S., Ermis, O., Mollering, H., onen, M., Schneider, T.: Privacy-preserving density-based clustering. In: Proc. ACM Asia Conf. Comput. Commun. Secur. (ASIACCS) (2021)
7. Cornuéjols, A., Wemmert, C., Gançarski, P., Bennani, Y.: Collaborative clustering: Why, when, what and how. Inf. Fusion (2018)
8. Costello, C., et al.: Geppetto: Versatile verifiable computation. In: Proc. IEEE Symp. Secur. Privacy (SP) (2015)
9. Davidson, S., Khanna, S., Milo, T., Panigrahi, D., Roy, S.: Provenance views for module privacy. In: Proc. 30th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst. (PODS) (2011)
10. Davidson, S., Khanna, S., Roy, S., Stoyanovich, J., Tannen, V., Chen, Y.: On provenance and privacy. In: Proc. 14th Int. Conf. Database Theory (ICDT) (2011)
11. De, I., Tripathy, A.: A secure two-party hierarchical clustering approach for vertically partitioned dataset with accuracy measure. In: Recent Advances in Intelligent Informatics (2014)
12. Dey, S., Zinn, D., Ludäscher, B.: Propub: Towards a declarative approach for publishing customized, policy-aware provenance. In: Proc. 23rd Int. Conf. Sci. Statist. Database Manag. (SSDBM) (2011)
13. Dwork, C., Smith, A., Steinke, T., Ullman, J.: Exposed! a survey of attacks on private data. Annu. Rev. Stat. Appl. **4**, 61–84 (2017)
14. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. Int. Conf. Knowl. Discovery Data Min. (KDD) (1996)
15. Everitt, B.S., Landau, S., Leese, M., Stahl, D.: Cluster Analysis. Wiley (2011)
16. Fisher, R.A.: Iris. UCI Machine Learning Repository (1988), https://doi.org/10.24432/C56C76

17. Fukunaga, K., Hostetler, L.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Trans. Inf. Theory (1975)
18. Gymrek, M., McGuire, A.L., Golan, D., Halperin, E., Erlich, Y.: Identifying personal genomes by surname inference. Science **339** (2013)
19. Hamidi, M., Sheikhalishahi, M., Martinelli, F.: Privacy preserving expectation maximization (em) clustering construction. In: Proc. DCAI (2019)
20. Jagannathan, G., Wright, R.N.: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In: Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min. (KDD) (2005)
21. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys **31**(3), 264–323 (1999)
22. Lin, X., Clifton, C., Zhu, M.: Privacy-preserving clustering with distributed em mixture modeling. Knowl. Inf. Syst. (2005)
23. Liu, J., Xiong, L., Luo, J., Huang, J.Z.: Privacy preserving distributed dbscan clustering. Trans. Data Privacy (2013)
24. Meng, X., Papadopoulos, D., Oprea, A., Triandopoulos, N.: Private two-party cluster analysis made formal and scalable. arXiv preprint arXiv:1904.04475v2 (2019)
25. Mohassel, P., Rosulek, M., Trieu, N.: Practical privacy preserving k-means clustering. In: Proc. PETS (2020)
26. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proc. IEEE Symp. Secur. Privacy (SP) (2008), https://doi.org/10.1109/SP.2008.33
27. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: Proc. 30th IEEE Symp. Secur. Privacy (SP). pp. 173–187 (2009), https://doi.org/10.1109/SP.2009.22
28. Nergiz, M.E., Atzori, M., Clifton, C.: Hiding the presence of individuals from shared databases. In: Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD) (2007), https://doi.org/10.1145/1247480.1247554
29. Power, J., Beresford, A.: Sok: Managing risks of linkage attacks on data privacy. In: Proc. Priv. Enhanc. Technol. (PoPETS) (2023), https://doi.org/10.56553/popets-2023-0043
30. Pyrgelis, A., Troncoso, C., De Cristofaro, E.: Measuring membership privacy on aggregate location time-series. Proc. ACM Meas. Anal. Comput. Syst. **4**(2) (2020), https://doi.org/10.1145/3392154
31. Rahman, M.S., Basu, A., Kiyomoto, S.: Towards outsourced privacy-preserving multiparty dbscan. In: Proc. IEEE 23rd Pac. Rim Int. Symp. Dependable Comput. (PRDC) (2017)
32. Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Density-based clustering in spatial databases: The algorithm gdbscan and its applications. Data Min. Knowl. Discov. (1998)
33. Wang, C., Xu, J., Tan, S., Yin, L.: Privacy-preserving cloud-edge collaborative k-means clustering model in iot. In: Yang, H., Lu, R. (eds.) Frontiers in Cyber Security, Commun. Comput. Inf. Sci., vol. 1992. Springer (2023), https://doi.org/10.1007/978-981-99-9331-4_44
34. Wu, W., Liu, J., Wang, H., Hao, J., Xian, M.: Secure and efficient outsourced k-means clustering using fully homomorphic encryption with ciphertext packing technique. IEEE Trans. Knowl. Data Eng. **33**(10), 3424–3437 (2021)
35. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Trans. Neural Networks **16**(3), 645–678 (2005)
36. Xu, X., Ester, M., Kriegel, H.P., Sander, J.: A distribution-based clustering algorithm for mining in large spatial databases. In: Proc. Int. Conf. Data Eng. (ICDE) (1998)
37. Yuan, C., Yang, H.: Research on k-value selection method of k-means clustering algorithm. J. **2**(2), 226–235 (2019)

reteLLMe: Design Rules for using Large Language Models to Protect the Privacy of Individuals in their Textual Contributions

Mariem Brahem[1,2], Jasmine Watissee[3,1], Cédric Eichler[4,1], Adrien Boiret[4,1], Nicolas Anciaux[1,2], and Jose Maria de Fuentes[5,1]

[1] Inria, Petscraft project-team, <firstname.lastname@inria.fr>
[2] Université Paris Saclay - Versailles, <firstname.lastname@uvsq.fr>
[3] Institut Polytechnique de Paris, <firstname.lastname@polytechnique.edu>
[4] INSA Centre Val de Loire, <firstname.lastname@insa-cvl.fr>
[5] Universidad Carlos III de Madrid, <josemaria.defuentes@uc3m.es>

Abstract. The advanced inference capabilities of Large Language Models (LLMs) pose a significant threat to the privacy of individuals by enabling third parties to accurately infer certain personal attributes (such as gender, age, location, religion, and political opinions) from their writings. Paradoxically, LLMs can also be used to protect individuals by helping them to modify their textual output from certain unwanted inferences, opening the way to new tools. Examples include sanitising online reviews (e.g., of hotels, movies), or sanitising CVs and cover letters. However, how can we avoid miss estimating the risks of inference for LLM-based text sanitisers? Can the protection offered be overestimated? Is the original purpose of the produced text preserved?
To the best of authors knowledge, no previous work has tackled these questions. Thus, in this paper four design rules (collectively referred to as *reteLLMe*) are proposed to minimise these potential issues. We validate these rules and quantify the benefits obtained in a given use case – sanitising hotel reviews. We show that up to 76% of at-risk texts are not flagged as such without fine-tuning. Moreover, classic techniques such as BLEU and ROUGE are shown to be incapable of assessing the amount of purposeful information in a text. Finally, a sanitisation tool based on *reteLLMe* demonstrates superior performance to a state-of-the-art sanitiser, with better results on up to 90% of texts.

Keywords: LLM · Privacy · Inference · Anonymisation

1 Introduction

Large Language Models (LLM) and related generative artificial intelligence techniques are on the rise. They are able to perform complex tasks such as video generation or speech synthesis, to name a few [2].

Despite their countless advantages, LLMs pose a serious threat to privacy by means of inferences [19,5]. Indeed, their ability to accurately predict sensitive

attributes (such as age, gender or political beliefs) of the author of a text that is believed to be benign has already been demonstrated [16]. This may lead to the complete deanonymization of the author of a piece of work that was intended to remain unknown, thus leading to undesired consequences.

Conversely, the very same inference capability of LLMs can be applied to mitigate the privacy threat. Previous works have already shown how LLMs can be converted into large-scale anonymizers, thus transforming a piece of data (say text, video, image, etc.) into another one that can reduce the precision of LLM-based inferences [17,18]. Some approaches have also considered the utility of the generated texts as a feature to preserve [4].

There are a number of motivating use cases for such a privacy-preserving use of LLMs. For example, in the context of online reviews, whether for hotels, products, or services, it is essential to protect the anonymity of reviewers to encourage honest and unbiased feedback. Here, a LLM can assist by sanitizing text to conceal Personally Identifiable Information (PII), ensuring consumer opinions are shared without fear of personal exposure. Similarly, in professional environments, such as job applications, ensuring that work-related documents such as reports or cover letters may not leak any highly sensitive information such as religious beliefs or political thoughts could help mitigate discrimination.

A great number of recent efforts revolve around using LLMs as privacy-preserving tools, such as [14,17], to name a few. Although they exhibit promising performance features, their use of LLMs does not follow any particular design criteria. This leads to undesired effects in text utility, user privacy or both. Therefore, it is essential to build those privacy-preserving LLMs in a sound manner, providing reliable evidence of their effectiveness.

To address these issues, this paper proposes a set of design rules (collectively referred to as $reteLLMe$) to build privacy-preserving LLMs and evaluate their efficiency. They provide solid grounds to achieve an optimal transformation of a piece of information while keeping a privacy-utility tradeoff. More specifically, the contributions of this work are as follows:

- We propose a novel problem statement by identifying three main underlying challenges, namely (1) the assessment on a realisic LLM-based attacker, (2) the sanitisation of the piece of information to limit these inferences and (3) the assessment of the utility of the sanitised output to maintain a threshold;
- We provide a set of design rules to address the above challenges;
- We show experimentally the benefit of following these rules, and conversely the impact of ignoring them, in the context of a hotel review sanitiser.

Paper organization. Section 2 introduces the problem statement. Section 3 introduces the proposed design rules in a case-agnostic fashion. Section 4 describes the application of these rules in the context of sanitising hotel reviews. Section 5 provides an experimental validation and comparison with the state of the art. Section 6 introduces the related work. Lastly, Section 7 concludes the paper and points out future work directions.

2 Problem Statement

This paper tackles the intricate task of text sanitisation through the utilization of LLMs to shield sensitive author attributes (e.g., age, gender, etc.). This Section introduces three underlying difficulties of this task (Sections 2.1, 2.2 and 2.3) and concludes with the problem formulation (Section 2.4).

2.1 Difficulty 1: Using LLMs for Privacy Risk Assessment

Recent studies have shown that off-the-shelf LLMs can infer personal information from texts [16]. However, utilizing pre-trained LLMs in a defensive manner to help individuals assess the risk of inference in their texts presents two main obstacles. Firstly, LLMs can sometimes produce inferences as accurate as a random guess, making them unreliable for privacy risk assessments without a mechanism to evaluate the inference likelihood. Secondly, fine-tuning LLMs has shown effectiveness in various contexts, such as reidentifying personal information from anonymized medical documents [18], highlighting the need for considering high-quality and diverse training datasets to accurately estimate privacy risks.

2.2 Difficulty 2: Assessing Text Utility after Sanitisation

Traditional free-text utility metrics like BLEU [11] and ROUGE [8] may be considered in text sanitisation (see e.g., recent preprint [17]). Such metrics are excellent for evaluating summaries by comparing n-gram co-occurrences between texts. However, they fail to differentiate between the loss of utility due to the modification of words and the destruction of relevant information. For example, consider the following fictitious hotel review and its sanitised version:

> [*Original text*] I went there with my husband Francis for the 3rd anniversary of our youngest child. The staff was delightful and the room clean.

> [*Sanitised text*] Family friendly. The staff was delightful and the room clean.

Although the sanitised version retains all the information essential for evaluating the hotel, an n-gram analysis would result in a low utility score with BLEU, of around 0.27. In addition, consider the identifying excerpt:

> [*Privacy-sensitive excerpt*] I went there with my husband Francis for the 3rd anniversary of our youngest child.

The BLEU value for this Privacy-sensitive excerpt is 0.57, which is surprisingly higher than the score for the sanitized text containing the more descriptive hotel review segment. This discrepancy highlights the challenge of accurately assesing the trade-off between privacy and utility.

2.3 Difficulty 3: Optimising the Overall Sanitisation Process

Designing a text sanitisation process that balances the preservation of content utility with a significant reduction in privacy risks is a multifaceted challenge. Indeed, the inference ability of potential adversaries equipped with fine-tuned LLMs and example datasets must be countered. Existing methods, like masking or direct removal of Personally Identifiable Information (PII), either tend to overly sanitise the text, diminishing its original utility, or retain enough information for sensitives to be inferred. For instance, Microsoft's Azure AI solution [1] can identify text segments that might expose PII and health identifiers (PHI). However, recent studies [16] indicate that simply blacking out these segments is not always effective against inferences made by current LLMs. Yet, these approaches often overlook the utility loss associated with de-identification [3].

2.4 Overall Problem Formulation

The challenge in developing an LLM-based text sanitisation tool is to strike a balance between preserving utility and mitigating privacy risks. To our knowledge, this intricate problem remains unsolved. The objective of this article is to provide a set of design rules to address this goal effectively. This complex task can be distilled into three building blocks:

- **Likelihood measure $\Lambda_{\mathcal{A}}$ for inferences**: assesses the validity of the inferred values for a given set $\mathcal{A}$ of sensitive attributes.
- **Utility measure $\mathcal{U}_{\mathcal{P}}$**: quantifies the utility of a text based on its alignment with a given purpose $\mathcal{P}$ for which the original text was created.
- **Sanitisation process $\mathcal{S}_{\Lambda_{\mathcal{A}},\mathcal{U}_{\mathcal{P}}}$**: transforms the original text into a sanitised text in order to reduce the likelihood of inferences while maintaining utility.

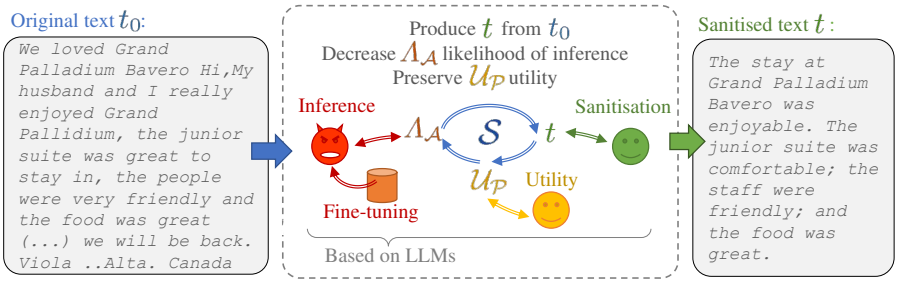


Fig. 1: Main building blocks of a text sanitisation process $\mathcal{S}$ using LLMs

Figure 1 illustrates the overall process and the building blocks for sanitising content using LLMs, from the user text t_0 in input (left) to the sanitised text t in output (right). Our goal is to provide essential guidelines for the design of such LLM-based systems that address the difficulties mentioned above.

3 *reteLLMe* Design rules

In order to address the problem, clear and effective *design rules* are required. This section introduces these design rules, collectively referred to as $reteLLMe$, organised around the three building blocks they address – inference (Section 3.1), utility (Section 3.2) and sanitisation (Section 3.3).

3.1 Inference Design Rules

The ability of LLMs to make inferences can be regarded as an adversary. The issues raised in Section 2.1 require the consideration of a realistic (i.e., sufficiently strong) adversary model suitable for the specific use scenario. An attacker model based on a generic LLM, as considered in recent works [16,17], may underestimate the attacker. This leads to a first practical design rule:

> **Design rule 1: Tailored Adversary LLM.** Avoid using generic attacker models, such as generic LLMs, as this may underestimate accuracy and privacy risks. Instead, employ tailored models such as fine-tuned LLMs.

On the other hand, the capability to infer using LLMs alone is not enough without evaluating the *likelihood* of these inferences. This leads to introduce:

> **Design rule 2: Well-Formed Likelihood Metrics.** The tool must incorporate a well-formed likelihood metrics $\Lambda_{\mathcal{A}}$ to predict the validity of guesses when truth values are unknown.

These design rules imply an attacker model where $\mathcal{RA}$ (Realistic Adversary) has access to a text t authored by a user u and is interested in a specific set of sensitive attributes $\mathcal{A}$. For each sensitive attribute $A{\in}\mathcal{A}$ (e.g., Age), D_A represents its domain, and $a_u{\in}D_A$ denotes its true value for u (e.g., the actual age of u). Using LLMs, $\mathcal{RA}$ produces a_t, the value it inferred from t for each a_u. To represent realistic threats, the adversary is assumed to have access to:

- a dataset D of pre-existing texts written by a set of users U, not including u, for which real values of sensitive attributes $a_{u'\in U}$ are known;
- a likelihood metric $\Lambda_{\mathcal{A}}$ which evaluates the accuracy of guesses about sensitive attributes (i.e., the probability that the inferred value a_t matches the true value a_u).

Designing $\Lambda_{\mathcal{A}}$ is challenging. The likelihood metric estimates the accuracy of each guess made by the attacker. Ideally, a guess represents the probability of its correctness. However, since the truth values of targeted users during the attack are unknown, this probability cannot be analytically computed.

A likelihood metric $\Lambda_{\mathcal{A}}$ is considered well-formed and satisfies design rule 2, if and only if it satisfies the following property: for u the author of text t, $\forall A \in \mathcal{A}$,

$\forall \epsilon \in [0,1], \quad \epsilon \to \frac{|\{t \in D : \Lambda_{\mathcal{A}}(a_t) > \epsilon \wedge a_t = a_u\}|}{|\{t \in D : \Lambda_{\mathcal{A}}(a_t) > \epsilon\}|}$ is a monotonically increasing function. It ensures that $\Lambda_{\mathcal{A}}$ can predict whether the probability of a guess being correct is above or below a given threshold. This capability allows an attacker to know whether the inference is plausible and it allows a sanitising tool to alert the user that their text is at risk.

3.2 Utility Assessment Design Rules

We advocate for the adoption of purpose-centric utility metrics alongside inference metrics. The objective is to identify and prioritize information aligned with the intended *purpose* of texts. This leads to the following rule:

> **Design rule 3: Purpose-Centric Utility.** The integration of purpose-centric utility metrics $\mathcal{U}_{\mathcal{P}}$, defined independently of privacy considerations and tailored to the specific purpose of the original text, is essential for maintaining the practical value of LLM-based sanitised outputs.

This guideline entails defining a purpose as a set $\mathcal{P}$ of purpose-related attributes, where each attribute $P \in \mathcal{P}$ represents a category of relevant information regarding the initial purpose for which the text was produced.When a text t produced by sanitising t_0 transmits information about an attribute linked to a purpose $\mathcal{P}$, other users can deduce a value from t. $\mathcal{U}_{\mathcal{P}}$ should evaluate the ability of t to convey *the same* information relevant to the purpose as t_0.

3.3 Sanitisation design rules

The process of sanitisation relies on LLMs to transform a text t_0 into a sanitised text t, aiming to reduce privacy risks while maintaining utility. We do not prescribe specific guidelines for the sanitisation process, whether it should be iterative, interactive, or otherwise.

The effectiveness of a sanitisation technique hence hinges on the independence between the purpose-centric attributes $\mathcal{P}$ and the sensitive attributes $\mathcal{A}$, leading to a fourth rule:

> **Design rule 4: Privacy-Utility Independance.** Sanitisation techniques must aim to decrease inference likelihood while retaining useful information. The efficiency of the sanitisation process is constrained by the degree of independence between privacy and utility metrics. In case where independence is lacking, residual privacy risks must be carefully evaluated and addressed.

The theoretical feasibility of a perfect sanitiser, which would fully preserve utility while nullifying privacy threats, relies on the independence of relevant information categories between sensitive attributes and purpose-centric ones. However, in practical scenarios where independence is not assured, different trade-offs

need exploration, and residual privacy risks must be considered. This underscores the importance of robust privacy risk assessment, as highlighted in Section 3.1.

4 Application of *reteLLMe* design rules

Demonstrating the suitability of the *reteLLMe* design rules requires instantiating them in a practical scenario. This section shows the instantiation of a sanitiser based on an LLM for sanitising hotel reviews. The scenario and dataset are described in Section 4.1. Then the design rules for inference, utility assesment and sanitisation are implemented in Section 4.2.

4.1 *"Hotel reviews sanitiser"*: scenario and dataset description

We consider a practical application scenario where a sanitisation tool based on $ChatGPT3.5$ is at stake – users enter their hotel review text and the tool rewrites the text to improve privacy while preserving the review utility. This tool could be integrated into commercial platforms such as Booking.com or Airbnb.

To validate our design rules, we use the PAN[6] dataset [13], which provides 4.160 hotel reviews written in English (see an example on left part of Figure 1). Each one has truth values of two attributes of its author – *gender* (male or female) and *age* ($[18, 24]$, $[25, 34]$, $[35, 49]$, $[50, 64]$ or $[65, xx]$).

4.2 Implementation and compliance to design rules

The inference values a_t for age (A) and g_t for gender (G) for a text t are produced using a specific prompt. The inference process involves fine-tuning $ChatGPT3.5$ using a random subset of the PAN hotel reviews dataset. Concerning likelihood values, they are also produced using specific prompts. All prompts are shared in our online repository as stated below.

In what comes to utility, we define a set of purpose-related attributes $\mathcal{P}$ that typically summarise hotel reviews, including general sentiment, specific problem noted, cleanliness, room quality and service standards. Each attribute in $\mathcal{P}$ is categorized into positive (*good*), negative (*bad*), or neutral/missing ($\perp$) values, reflecting its impact on the overall assessment of hotel performance. For each $P \in \mathcal{P}$, we define a binary utility, which is 1 if $ChatGPT3.5$ provides the same answer for t and t_0 (the latter being a non-null value), and 0 otherwise. The overall utility $\mathcal{U}_{\mathcal{P}}$ is computed as the average of these utilities. Concerning sanitisation, $ChatGPT3.5$ is instructed to eliminate textual elements that could reveal the reviewer's age or gender. Subsequently, the anonymised text is rewritten in a neutral tone to mitigate unintended biases.

[6] PAN is an annual competition [12] that provides datasets for different tasks, including author profiling. We are using the 2014 dataset which is the most comprehensive provided for our use. PAN also provides the accuracy achieved by the winners, which will be used for comparison.

All these decisions are in line with design rules: we apply fine-tuning (Design rule 1); the likelihood of each inference is also self-evaluated by $ChatGPT3.5$, and we show experimentally in the next section that the resulting likelihood metric is well-formed (Design rule 2); we score texts depending on their information across purpose-related attributes (Design rule 3) and we balance privacy and utility in sanitized texts (Design rule 4).

5 Assessment of *reteLLMe* design rules

This section presents the experimental results on the proposed $reteLLMe$ design rules. First, the methodology and experimental settings are discussed in Section 5.1. Afterwards, we use the same order as in the previous sections – Section 5.2 focuses on the inference process, Section 5.3 on utility computation, Section 5.4 on the sanitisation effectiveness. Lastly, Section 5.5 discusses the results.

5.1 Experimental settings

Since the attacker model involves fine-tuning which may lead to variations depending on the training dataset, we randomly partition the dataset in four. Each partition is randomly split into two categories, training (H_{840}, approximately 80% of texts) and tests (the remaining 20%). Experiments are run four times on each dataset. Reported results are the average of all experiments.

To generate our prompts, we used well-known techniques (e.g., "Let's play a game...") to force ChatGPT[7] to perform undesired actions [7,16]. To fine-tune ChatGPT, we used OpenAI's dedicated API [10]. To foster further research, our experimental scripts and prompts are publicly released[8].

5.2 Validating reteLLMe measure for inference and likelihood

The goal of this section is to validate the inference module by (i) evaluating the impact of fine-tuning ChatGPT and confirming the strength of the realistic attacker (to assess Design rule 1) and (ii) validating the proposed likelihood metric (to assess Design rule 2).

Attacker definitions. In accordance with Design rule 1, our experiment considers a (realistic) strong attacker relying on a fine-tuned version of ChatGPT as described above. For comparison, this *strong attacker* is compared to a *weak attacker* that behaves similarly but relies on an off-the-shelf ChatGPT.

[7] We used $ChatGPT3.5$.
[8] GitHub repository to be added after acceptance

Fine-tuning - Design rule 1. Figure 2 shows the accuracy of the strong attacker and its weak version with respect to a random guess and the best scores in each category of the PAN competition. The random guess ("baseline") has a score of 0.5 for gender as there are only two options (male/female), and 0.2 for the age as there are five ranges (see Section 4.1). Thus, the *total* baseline accuracy is 0.1 as the product of the two.

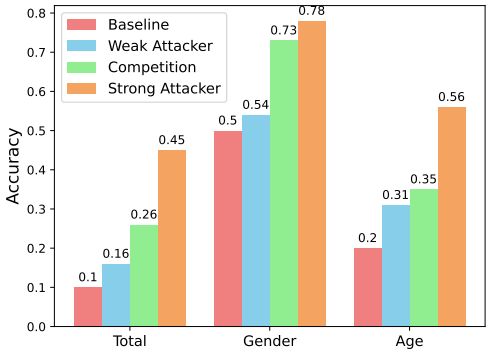
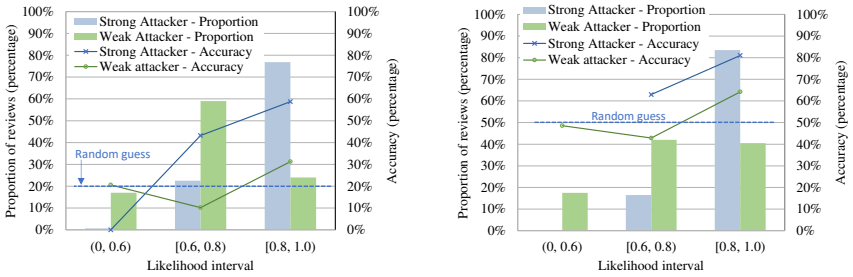


Fig. 2: Accuracy of the weak and strong $reteLLMe$-compliant attackers

Figure 2 shows a significant improvement in the total accuracy from 0.16 to 0.45 with fine-tuning, surpassing both the baseline and the competition results. Furthermore, the comparative analysis between age and gender reveals that the gender category benefits more from fine-tuning than the age category. This could be explained by the differences in the complexity of inferring age, which involves multiple categories, versus gender, which is classified as female or male.

Well-formed likelihood - Design rule 2. Figure 3 shows the number of inferences and their average accuracy as a function of their likelihood range. Each attacker provides its own likelihood, so the inferences are partitioned twice according to both. For each likelihood range, the bars show the proportion of inferences (percentage of reviews, left axis) and the curves show their average accuracy (right axis). Note that when no review lies in a likelihood interval, the value of the curve representing accuracy cannot be computed (as in the case of (0,0.6) for gender with the strong attacker).

Accuracy of the strong attacker for age and gender shows a correlation with likelihood (Pearson value of 0.99 for age and 0.96 for gender), as opposed to the weak attacker (age: 0.33, gender: 0.56). In fact, for the strong attacker, the accuracy increases monotonically for increasing likelihood ranges, showing that the likelihood metric is in this setting a well-formed metric (in the sense of Section 3.1) that satisfies *Design rule 2.*



(a) Age (b) Gender

Fig. 3: Accuracy of inference based on likelihood (original texts)

Underestimation of risks - Design rule 1. Beyond the well-formedness, Figure 3 illustrates the distribution of text accross different likelihood levels. A text is at risk when it falls within a likelihood range where the average accuracy significantly exceeds random guessing. Here, in reality, all texts are at risk. The (realistic) strong attacker, adhering to *reteLLMe* design rules, achieves a likelihood greater than 0.6 for each text, with an average accuracy well above random guessing, even within the $[0.6, 0.8)$ likelihood interval. On the contrary, the weak attacker fails to identify texts with likelihood below 0.8. This discrepancy leads to a significant underestimation of risks: 76% of texts (for age) and 59.5% (for gender) with high inference risks would not be flagged as such.

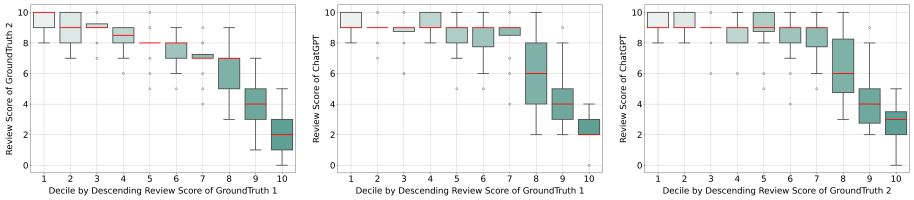
5.3 Validating *reteLLMe* measure for utility

This section validates the purpose-centric *reteLLMe* utility measure presented above (see Section 4.2). It first compares the responses obtained using this measure with those provided by humans. It then shows how BLEU and ROUGE behave and concludes on the importance of *Design rule 3*.

Automated purpose-related utility - Comparison with humans. We assign to each review a score out of 10. The score is calculated as the sum of each of the five purpose-related attributes, by assigning 2, 1, and 0 points to each "good", "neutral" and "bad" value respectively. This score is therefore not a measure of the utility of the review (a negative review can be very useful), but simply a way of ranking the reviews from the most positive to the most negative.

Figure 4 shows the distribution of scores for each of the humans and ChatGPT responses. Each box represents 10% of reviews, from the most positive to the most negative. These three curves show a strong similarity between the humans themselves and between the humans and ChatGPT. This leads to Pearson correlations of 0.8 and 0.82 between each human and ChatGPT. However, it should be noted that the variations in ChatGPT are less uniform than those

between humans. Therefore, although imperfect, ChatGPT is a good source of utility in the absence of ground truth.



(a) Ground truth 1 vs Ground truth 2 (b) Ground truth 1 vs ChatGPT (c) Ground truth 2 vs ChatGPT

Fig. 4: Human and ChatGPT-based evaluation of purpose-related attributes

BLEU/ROUGE against purpose-related utility - Design rule 3. To analyse the suitability of BLEU and ROUGE metrics, we examine their alignment with purpose-related utility metric by applying these measures to original hotel reviews (t_0) and sanitised versions (t) and comparing obtained scores with utility preservation according to our proposal.

Figure 5 presents the distribution of BLEU and ROUGE scores, categorized into quintiles with increasing utility preservation. Utility preservation for each sanitized text t compared to original text t_0 is determined by analysing $ChatGPT3.5$'s responses to a purpose-related questionnaire using both texts. This involves calculating binary utility $\mathcal{U}_P$ for purpose-related attribute based on $ChatGPT3.5$'s responses (which is 1 is same answer is provided, and 1 otherwise) and averaging these values for the five attributes to derive utility preservation (i.e., 1 means fully preserved with same answers to the five purpose-related questions, 0 means answers are all different).

Our results show a significant decorrelation between BLEU/ROUGE scores and utility preservation. Hence, they inadequately assess the purposeful information conveyed in texts, highlighting the negative repercussion of disregarding *Design rule 3*.

5.4 Sanitisation effectiveness

We compare our method to the two settings of the anonymiser proposed by Azure, Azure (All entities) and Azure (Three entities) [1]. There are three issues to consider – whereas after sanitisation the inference likelihood is effectively reduced, the inference accuracy is also decreased and the utility is preserved, as needed to satisfy *Design rule 4*.

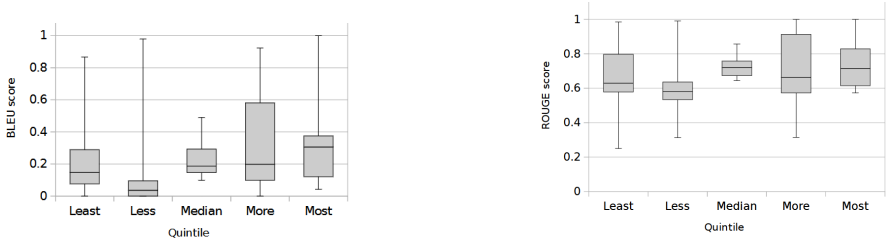
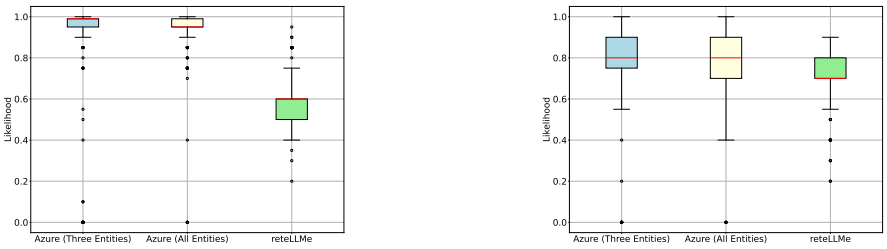


Fig. 5: BLEU/ROUGE scores of sanitised texts ordered by purpose-centric utility

Decreasing likelihood. Figure 6 shows the distribution of the inference likelihood for both attributes in the three considered methods. Intuitively, lower likelihood values are preferred from a privacy perspective.



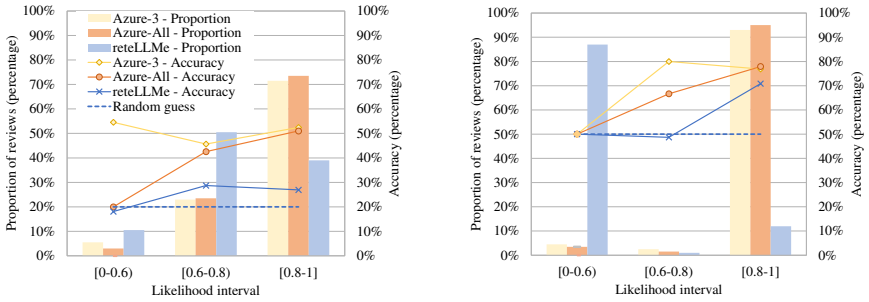
(a) Gender (b) Age

Fig. 6: Inference likelihood after sanitisation

Figure 6a illustrates the inability of Azure to sanitise gender-related data in both settings, resulting in median likelihood scores of 0.99 and 0.95. Unsurprisingly, the setting removing less data, "Three entities", exhibits the highest likelihood score. On the contrary, *reteLLMe* is significantly more effective to protect this attribute, leading to a median likelihood of 0.6. A similar situation happens for the attribute age (Fig. 6b). Thus, our module outperforms both settings of Azure for the protection of both age and gender attributes.

Decreasing accuracy. Figure 7 shows the distribution of accuracy values for the three methods. Recall that random guess thresholds are different for age (which is 0.2) and gender (being 0.5). Intuitively, lower likelihood scores lead to smaller accuracy values. Interestingly, *reteLLMe* exhibits good behavior for the highest level of likelihood. Thus, an average accuracy of 0.27 and 0.71 is reached for the age and gender, respectively. For lower likelihood ranges, *reteLLMe* accuracy is closer to a random guess.

As a matter of fact, the amount of texts that remain at risk after sanitisation is largely different. Remarkably, for both attributes and both Azure variants, more than 90% of reviews remain at risk, i.e. belongs to likelihood intervals with an average accuracy significantly more than random guessing (up to 52% average accuracy for age and 78% for gender). When *reteLLMe* is applied, only 11% of reviews are at risk with regard to gender.



(a) Age (b) Gender (same legend)

Fig. 7: Accuracy of inference based on likelihood (after sanitisation)

Utility preservation. Beyond reducing the likelihood of inferences, it is also necessary to ensure that the utility is preserved. Figure 8 shows the distribution of utility preservation across the three methods at stake. The three methods lead to a substantial utility preservation, as the highest amount of records count on the biggest utility preservation figures. Indeed, *reteLLMe*, Azure 3 entities, and Azure all entities preserve between 80% and 100% utility of 71.6%, 70,5%, and 65.6% of texts respectively. Overall and as expected, the "3 entities" setting outperforms the "all entities" setting with regard to utility preservation.

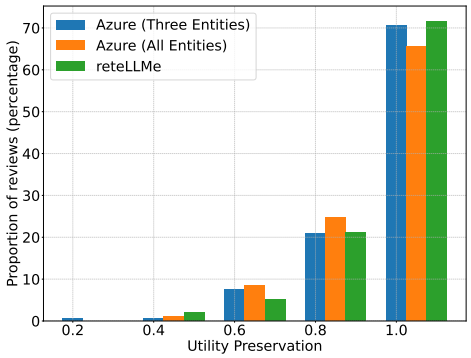


Fig. 8: Distribution of utility preservation

5.5 Discussion

Our experimental results are merely limited to the use-case of hotel reviews with a limited test dataset (4 experiments with 200 texts per experiment). They are however valid to confirm the proposed guidelines.

First, they confirm that LLMs such as ChatGPT can be used as privacy-enhancing tools in an effective manner that outperforms industrial state of the art anonymisers in term of both risk minimization and utility preservation. More importantly, they demonstrate the importance of our design rules.

Indeed, they illustrate the effect of considering a strong adversary by showing how fine-tuning impacts ChatGPT's inference ability. They demonstrate how generic adversaries may severely underestimate privacy risks.

As opposed to the state-of-the-art, they shown that purpose-centric utility metrics are a differentiating factor. Specifically, it is shown that generic metrics such as ROUGE or BLEU may be wholly decorrelated to the amount of purposeful information present in a text.

Our results, however, are limited in that Design rule 4 asked for a sanitisation procedure that decreases the inference likelihood while preserving utility. Nevertheless, our input to ChatGPT does not include the inference likelihood. Our results confirm that even without that input, the inference likelihood and accuracy are severely decreased while the utility is preserved. We argue that this phenomenon cannot be extrapolated to any use case, as it may be due to the very nature of hotel reviews. Nevertheless, our assessment is comprehensive enough to confirm that even the most simplified sanitiser observing this design rule is effective enough.

6 Related Work

The use of LLMs as privacy-enhancing technologies has already attracted some research attention. Indeed, an increasing amount of papers have been produced in the last years. Interested readers may refer to a systematic literature review by Sousa *et al.* [15]. In a nutshell, LLMs seem to be a suitable technology considering the challenges posed by text anonymization, due to the unbounded nature of information related to individuals [9].

[16] was the first study to highlight the critical privacy concerns posed by LLMs beyond the commonly discussed issues related to data memorization. They show that LLMs are able to identify personal data at an unprecedented scale and emphasizes the need for new anonymization techniques to counteract such evolving threats. While the authors effectively illustrate the problem the of inferring personal data from text, they do not propose any solution. Our research builds upon their findings by not only considering these issues but also proposing new guidelines to mitigate the risk of such inferences. Moreover, contrary to our guidelines, they count on a generic attacker – their LLM is not fine-tuned.

A follow-up work by the same authors [17] marks a significant advancement in this domain. They propose an evaluation framework that leverages the capabilities of LLMs for text anonymization. It employs a multiple round process

where a LLM adversary analyzes the text for private attribute inference, followed by an anonymizing LLM that modifies the text to obscure identifiable information. Furthermore, this framework introduces a binary "certainty" scoring system discriminating inferences depending on whether they rely on statistical bias or directly identifiable information within texts. We consider that this "certainty" score serves as a metric of inference likelihood. However, [17] does not validate its relation to accuracy. Since statistical analysis may provide accurate guesses, it is not immediate that certainty is a good predictor of accuracy.

Beyond the inference likelihood, Staab *et al.* consider utility metrics such as BLEU and ROUGE. Complementarily, they use a "judge" prompt that assesses anonymized texts across three dimensions – readability, meaning, and hallucination. Interestingly, the "meaning" dimension assesses semantic proximity that could be purpose-centric. Since it takes into account *all* information, we argue that it exhibits the same limitations as those discussed in Section 2.2 and can never be independent from privacy considerations. This is supported by the observation that BLEU, ROUGE, and judge exhibit the same trends [17].

Another relevant study is [4]. The authors introduce the concept of self disclosure abstraction that allows paraphrasing personal disclosures into more general terms without losing their communicative value, reducing privacy risks while preserving the overall utility (e.g., "Im 16F" to "I'm a teenage girl".) Their methodology involves a fine-tuning strategy to identify instances of self-disclosures which confirms the importance of considering a realistic attacker model as highlighted in our guidelines. However, their approach does not work properly for sensitive attributes which are not directly mentioned in the text, as [16] already proved.

[18] addresses the issue of the de-identification of clinical reports to facilitate data access for research purposes while ensuring patient privacy using the CamemBERT model, a BERT variant specially crafted for French texts. This approach aligns partially with our proposed guidelines. They count on a well-formed inference likelihood metric. Moreover, their attacker model is concrete enough due to fine-tuning. However, it does not include the notion of utility.

Our work distinguishes itself from recent research by proposing a number of guidelines that have been partially overlooked by previous efforts, as discussed above. To the best of our knowledge, this is the first effort in this direction. Our work has also illustrated which is the impact of not following these guidelines.

7 Conclusion

LLMs have already been shown to be effective to both anonymise and de-anonymise texts. This dual nature gives them an unprecedented ability to be used as privacy-enhancing technology. However, previous attempts have failed to propose such an usage considering the common pitfalls in text utility, inference assessment and sanitisation effectiveness. In this vein, this work has proposed $reteLLMe$, a collection of design rules in this regard. Our assessment in the context of protecting hotel reviews has not only shown the convenience of the proposed rules, but also the negative consequences of disregarding them.

Future work will focus on exploring the suitability of these rules in other contexts. In this vein, the design of well-formed and generalizable purpose-centric utility metrics is envisioned as a critical issue. On the other hand, exploring the impact of LLM-based threats such as privacy leakages [6] in this privacy-enhancing usage is another interesting direction.

Acknowledgement

This work was supported by the French grant iPoP PEPR (ANR-22-PECY-0002). Jose Maria de Fuentes has been partially supported by the Spanish National Cybersecurity Institute (INCIBE) grant APAMciber within the framework of the PRTR funds, financed by the European Union (Next Generation). Jose Maria de Fuentes has also received support from UC3M's Requalification programme, funded by the Spanish Min. de Ciencia, Innovacion y Universidades with EU recovery funds (Convocatoria de la UC3M de Ayudas para la recualificación del sistema universitario español para 2021-2023, de 1 de julio de 2021).

References

1. Azure: What is Azure AI Language? https://learn.microsoft.com/en-us/azure/ai-services/language-service/overview (July 2023), [Online; accessed 01-April-2024]
2. Bai, J., Men, R., Yang, H., Ren, X., Dang, K., Zhang, Y., Zhou, X., Wang, P., Tan, S., Yang, A., et al.: Ofasys: A multi-modal multi-task learning system for building generalist models. arXiv preprint arXiv:2212.04408 (2022)
3. Berthelier, G., Boutet, A., Richard, A.: Toward training nlp models to take into account privacy leakages. In: 2023 IEEE International Conference on Big Data (BigData). pp. 4854–4862. IEEE (2023)
4. Dou, Y., Krsek, I., Naous, T., Kabra, A., Das, S., Ritter, A., Xu, W.: Reducing privacy risks in online self-disclosures with language models. arXiv preprint arXiv:2311.09538 (2023)
5. Kandpal, N., Pillutla, K., Oprea, A., Kairouz, P., Choquette-Choo, C., Xu, Z.: User inference attacks on llms. In: Socially Responsible Language Modelling Research (2023)
6. Kim, S., Yun, S., Lee, H., Gubri, M., Yoon, S., Oh, S.J.: Propile: Probing privacy leakage in large language models. Advances in Neural Information Processing Systems **36** (2024)
7. Li, H., Guo, D., Fan, W., Xu, M., Huang, J., Meng, F., Song, Y.: Multi-step jailbreaking privacy attacks on ChatGPT. In: Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 4138–4153. Association for Computational Linguistics (2023)
8. Lin, C.Y.: Rouge: A package for automatic evaluation of summaries. In: Text summarization branches out. pp. 74–81 (2004)
9. Lison, P., Pilán, I., Sánchez, D., Batet, M., Øvrelid, L.: Anonymisation models for text data: State of the art, challenges and future directions. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). pp. 4188–4203 (2021)

10. OpenAI: Openai documentation, https://platform.openai.com/docs/guides
11. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics. pp. 311–318 (2002)
12. Pardo, F.M.R., Rosso, P., Koppel, M., Stamatatos, E., Inches, G.: Overview of the author profiling task at PAN 2013. In: Forner, P., Navigli, R., Tufis, D., Ferro, N. (eds.) Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013. CEUR Workshop Proceedings, vol. 1179. CEUR-WS.org (2013), https://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-RangelEt2013.pdf
13. Rangel Pardo, F., Rosso, P., Chugur, I., Potthast, M., Trenkmann, M., Stein, B., Verhoeven, B., Daelemans, W.: Overview of the 2nd author profiling task at pan 2014. CEUR Workshop Proceedings **1180**, 898–927 (2014)
14. Song, Y., Zhang, J., Tian, Z., Yang, Y., Huang, M., Li, D.: Llm-based privacy data augmentation guided by knowledge distillation with a distribution tutor for medical text classification. arXiv preprint arXiv:2402.16515 (2024)
15. Sousa, S., Kern, R.: How to keep text private? a systematic review of deep learning methods for privacy-preserving natural language processing. Artificial Intelligence Review **56**(2), 1427–1492 (2023)
16. Staab, R., Vero, M., Balunović, M., Vechev, M.: Beyond memorization: Violating privacy via inference with large language models. arXiv preprint arXiv:2310.07298 (2023)
17. Staab, R., Vero, M., Balunović, M., Vechev, M.: Large language models are advanced anonymizers. arXiv preprint arXiv:2402.13846 (2024)
18. Tannier, X., Wajsbürt, P., Calliger, A., Dura, B., Mouchet, A., Hilka, M., Bey, R.: Development and validation of a natural language processing algorithm to pseudonymize documents in the context of a clinical data warehouse. Methods of Information in Medicine (2024)
19. Zhang, X., Xu, H., Ba, Z., Wang, Z., Hong, Y., Liu, J., Qin, Z., Ren, K.: Privacyasst: Safeguarding user privacy in tool-using large language model agents. IEEE Transactions on Dependable and Secure Computing (2024)

Plausible Deniability of Redacted Text

Vaibhav Gusain[1][0000−0002−7008−5201] and Douglas Leith[1][0000−0003−4056−4014] ⋆

Trinity College Dublin, Ireland

Abstract. Providing privacy for natural language text data remains a largely open problem, despite its great practical importance. The current state of the art is manual redaction of sensitive words such as names, addresses etc. In this paper we propose viewing a corpus of text as a probability distribution over sequences of words. A sentence is then one realization from this distribution and redacting words changes the probability distribution. We use the Renyi-divergence divergence as a measure of the distance between two redacted datasets. We show that if enough words are redacted then sensitive redacted text can be made be statistically indistinguishable from non-sensitive redacted text. This can be used to develop efficient redaction strategies, that minimise the amount of redaction while meeting a privacy target.

Keywords: Data Privacy · Natural Language Processing · Text Sanitization.

1 Introduction

Training of neural nets such as large language models requires the availability of natural language text training data. In this paper we revisit the question of how to sanitise sensitive text data so that it can be used for model training while preserving privacy. We introduce a new approach for quantitatively estimating the privacy gain from text redaction and demonstrate its usefulness on a wide range of datasets. This can be used to develop efficient redaction strategies, that minimise the amount of redaction while meeting a privacy target. The approach is closely related to differential privacy, but differs in several respects that are important for text data.

There are two main approaches to enhancing privacy when training machine learning models. These approaches are complementary, and both can be used together. One is to add noise to the gradient updates used during training, e.g. see DP-SGD (1) (17) and related work. The other is to sanitise the training data itself. Here we focus on the latter.

When the training data is numeric then the addition of appropriate noise, e.g. Laplacian noise scaled proportionally to the differential privacy (DP) "sensitivity" of the data, can be used to enforce differential privacy guarantees (7).

⋆ This work was supported by Science Foundation Ireland grant 16/IA/4610.

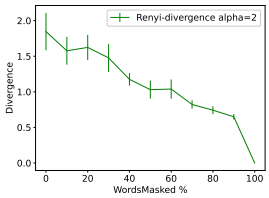


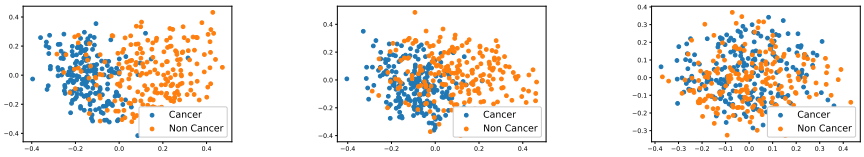
Fig. 1: Measured Renyi vs random redaction level for Medal dataset.

While it is tempting to apply this approach to text by mapping the text to a numeric embedding vector, adding noise, and then mapping back to text (8; 5; 20), this creates a host of unpleasant issues. For example, the nature of the mapping from text to vectors (which texts are mapped to vectors near or far away from one another) directly affects the impact of added noise, and so privacy. However, this is poorly understood, especially for modern embedding approaches based on neural networks. Another deeply problematic issue is that the words in a sentence tend to be correlated in complex ways, making any privacy approach based on individual words tend to overestimate the privacy gained.

In practice, the most popular approach for sanitising text data is redaction i.e. replacing selected words with an uninformative mask token. This is, for example, already widely used to remove personally identifying information (PII), e.g. names and addresses, from documents (12). However, other aspects of the text data can also be sensitive. For example, the text data may reveal sexual/gender traits, political preferences, health-related information/concerns, social and racial characteristics, etc of the user population from which the data was gathered. Textual style can also act as a personal fingerprint facilitating de-anonymisation and membership attacks against neural nets trained on this data.

Protecting privacy is especially challenging with text data because simply redacting specified keywords is rarely enough: the surrounding context can easily continue to reveal sensitive information (3). For example, in the sentence "I am diagnosed with cancer. I have to go to St Lukes for chemotherapy and will probably lose my hair" redacting "cancer" and "chemo" is not sufficient to conceal the cancer diagnosis if St Lukes is known to be a cancer care hospital. If "St Lukes" is also redacted, the combination of "diagnosis" and "lose my hair" is still enough to indicate a cancer diagnosis with high probability.

In summary, there is an urgent need more effective methods for improving the privacy of text data. Given the challenging nature of natural language text, it is probably too much to hope for theoretical guarantees but that should not stop us from trying to develop useful methods motivated by theoretical analysis. In this paper we take a step in that direction. We use redaction to add "noise" to text and by staying within original text domain thereby avoid most of the issues with numerical embeddings, and by working with text corpuses rather



(a) No redaction (b) 20% redaction (c) 40% redaction

Fig. 2: Illustrating the increasing overlap between the semtence embeddings for cancer and non-cancer text from the Medal dataset as the level of redaction is increased. SentenceBERT embeddings are projected to two dimensions using PCA, random redaction is used.

than individual words or sentences we can accommodate the word correlations within sentences.

1.1 Our approach

A dataset $\mathcal{D}$ is a collection of items. Each item is a sequence $x = (x_1, ..., x_{|x|})$ of words x_t belonging to a fixed vocabulary and with length $|x| \leq N$, N being the maximum admissible length. A redaction policy $\pi_p(x)$ maps sequence x to a new sequence where some words have been redacted i.e. replaced by an uninformative mask token `MASK`. We will assume that every redaction policy is parameterised by a parameter p taking a value between 0 and 1 such that when $p = 0$ then no words are redacted, when $p = 1$ then every word is redacted. For example, the uniform random $\pi_{rand,p}(x)$ redaction policy redacts each word in sequence x with probability p. Alternatively, we might rank the words in our vocabulary by their sensitivity and redact the top p fraction of these.

Each item x in a dataset is a random draw from a probability distribution $P(x)$ over sequences of words. After redaction, each element x is mapped to a new sequence $redact(x)$ and the redacted dataset becomes a sample from probability distribution $redact(P)$. We measure the distance between two redacted datasets $redact(\mathcal{D}_0)$ and $redact(\mathcal{D}_1)$ by the smallest value of $\epsilon \geq 0$ such that $\tilde{P}_0(y) \leq e^{\epsilon}\tilde{P}_1(y) + \delta$ and $\tilde{P}_1(y) \leq e^{\epsilon}\tilde{P}_0(y) + \delta$ where $\tilde{P}_0 := redact(P_0)$ is the probability distribution over token sequences in dataset $redact(\mathcal{D}_0)$, $\tilde{P}_1 = redact(P_1)$ in dataset $redact(\mathcal{D}_1)$ and y is any redacted sequence of words with length $|y| \leq N$.

This distance measure is similar to that used in (ϵ, δ)-differential privacy but with the difference that the set of neighbouring databases now consists of the single database $redact(\mathcal{D}_0)$ rather than all databases differing from $redact(\mathcal{D}_1)$ by a single element. When ϵ, δ are sufficiently small, the publication of private dataset $redact(\mathcal{D}_1)$ then only provides an attacker with limited new information over and above that already available from the public dataset $\mathcal{D}_0$. That is, we gain privacy in the sense of *indistinguishability* between the $redact(\mathcal{D}_0)$ and $redact(\mathcal{D}_1)$ datasets. It will prove convenient to work in terms of the Renyi-divergence

$D_\alpha(\tilde{P}_0||\tilde{P}_1)$ to calculate the distance between the datasets. We then convert this to an (ϵ, δ)-privacy guarantee using equations (2) and (3). See Section-4.1 and Section-4.2 for more details.

We assume the availability of a "safe" dataset $\mathcal{D}_0$ e.g. a public dataset that is suitably diverse and non-sensitive. Applying redaction policy π_p to both the sensitive dataset $\mathcal{D}_1$ and the safe dataset $\mathcal{D}_0$ then we expect that distance ϵ between the datasets decreases as the level of redaction increases, the distance becoming zero after redaction with $p = 1$.

Figure 1 illustrates this for the Medal dataset of medical records (see below for further details). The original dataset is split into a dataset $\mathcal{D}_1$ of cancer patients and a dataset $\mathcal{D}_0$ of non-cancer patients. Random redaction is used. The figure shows the measured Renyi-Divergence $D_{max}(P_0||P_1)$ between the empirical probability distributions P_0 and P_1 induced by $\mathcal{D}_0$ and $\mathcal{D}_1$ as the level of redaction is varied. As expected, it can be seen that the divergence decreases as the amount of redaction increases i.e. the two datasets become more similar.

Figure 2 illustrates this behaviour more visually. Redacted sentences are mapped to embedding vectors using SentenceBERT (15), the vectors are then projected onto to dimensions using PCA and shown as a scatter plot. It can be seen that without redaction the sentence embeddings have little overlap but as the level of redaction increases the overlap between the embeddings increases, indicating that distinguishing between the two datasets is becoming harder.

We make the following observations. (i) Redaction sanitizes the text data itself (rather then vector embeddings) and so yields a sanitized dataset that can be used for training ML models that take word sequences as input. (ii) By working in terms of the probability distribution over word *sequences* we take account of the correlation between the words in a sentence (the word context) and the associated potential for leakage of sensitive information. (iii) Sanitising the dataset is akin to local differential privacy i.e. the input to a query is perturbed to ensure privacy rather than the output of the query being perturbed. (iv) As the distance ϵ between the sanitized dataset and the safe dataset decreases, privacy increases but of course we expect utility to decrease (the added value of the new dataset decreases).

2 Related work

The existing literature on enhancing the privacy of text data can be roughly categorised as follows:

Redacting PII. Much of the literature on text redaction has focussed on redacting personally identifying information (PII). For example, (12) uses an ensemble of deep learning methods to detect and redact PII information from the medical notes of the patient, (2) considers the discovery of names, home towns, etc in student discussion boards. Other recent work includes (6) (21) (16).

Word-Level DP. Word-level DP approaches map an individual word to a vector embedding, add noise and then either map back to a new word or use the

noisy embedding directly. See e.g. (8; 20; 5). A typical choice of vector embedding is Glove (14). The choice of vector embedding has to made up front and its properties affect the privacy gained[1] in ways that remain very poorly understood. Words are discrete quantities and the impact of quantisation when mapping from vectors back to words also remains poorly understood. The DP "database" is a sentence and the words are the database entries. The DP guarantee (modulo concerns regarding the embedding already noted) therefore relates to insensitivity to an individual word in a sentence. However this DP analysis ignores correlations between the words in a sentence and so can underestimate the information release. The impact of correlations on DP is well known and was first noted by (9). See (10) for further discussion on the deficiencies of word-level DP.

3 Threat model

We consider the use of natural language text datasets for training machine learning models. We assume that the sensitive dataset itself is stored securely, but the sanitized/redacted dataset is publicly released. The main threat we consider is that that the sanitized dataset may be used to infer sensitive user traits e.g sexuality, health conditions, political preferences. This is a particularly topical concern since the development of LLMs is currently being driven by companies with a commercial interest in identifying user traits e.g for use in targeting adverts or other services. We assume that PII (names, addresses etc) has been removed, there being many existing techniques for this, see e.g. (6) (21) (16)

4 Preliminaries

The Renyi-divergence of order $\alpha > 1$ between two probability distributions P_0 and P_1 on sample space Y is (13):

$$D_\alpha(P_0||P_1) = \frac{1}{\alpha - 1} \log \int_Y P_0(x)^\alpha P_1(x)^{1-\alpha} dx \tag{1}$$

and similarly for $D_\alpha(P_1||P_0)$. When $\alpha = 1$ the Renyi-divergence equals the KL-divergence (11).

We say that two probability distributions P_0 and P_1 are (ξ, ρ)-zero-concentrated differentially private when:

$$D_\alpha(P_0\|P_1) \leq \xi + \rho\alpha \tag{2}$$

for all $\alpha > 1$. In the differential privacy literature P_0, respectively P_1, is the probability distribution induced by a randomised mechanism $\mathcal{M}$ applied to dataset

[1] Adding noise to an embedding perturbs it to nearby words, the way in which words are mapped to be close together (or far apart) therefore directly affects the output of the word-level DP sanitisation process.

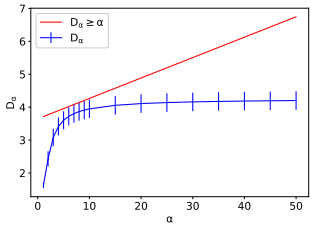


Fig. 3: Divergence vs α for non-redacted cancer and non-cancer text from Medal medical dataset.

$\mathcal{D}_0$, respectively $\mathcal{D}_1$ (4). Inequality (2) is then required to hold for all neighbouring datasets $\mathcal{D}_0$, $\mathcal{D}_1$, where datasets are neighbours if they differ by a single element. However, here we will consider other choices for P_0 and P_1. Specifically, they will be the distributions from which two datasets $\mathcal{D}_0$ and $\mathcal{D}_1$ are sampled.

When P_0 and P_1 are (ξ, ρ)-zCDP then from the proof of Lemma 21 in (4),

$$P_1(x) \leq \exp(\epsilon) P_0(x) + \delta \tag{3}$$

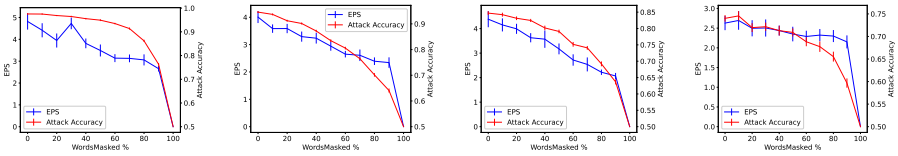
for every $\delta > 0$ where $\epsilon = \xi + \rho + 2\sqrt{\rho \log \frac{1}{\delta}}$. We will use (3) to map from Renyi-divergence curves to an (ϵ, δ) privacy guarantee.

4.1 Estimating Renyi-Divergence

To estimate the Renyi-divergence between two datasets we extend the estimator of (13), which is observed to scale well for high dimensional data. We updated the estimator to handle duplicate word-sequences, since these can become common following redaction[2]. In addition, each word sequence is mapped to a vector embedding[3] X_i. These are then fed to the estimator to calculate the Renyi-divergence. We use boot-strapping to calculate confidence intervals for the estimate. Namely, we sample with replacement n times from $redact(\mathcal{D}_0)$ and $redact(\mathcal{D}_1)$, estimate $D_\alpha(P_0||P_1)$ is calculated for each sample and then the mean and standard deviation of these n estimates calculated. We select n by calculating the mean and standard deviation vs n and selecting a value large enough that these are convergent. The mean of the estimated Renyi divergence is shown in our plots with the standard deviation indicated by error bars.

[2] See Appendix https://anonymous.4open.science/r/appendix_repo-F4CC for more details.

[3] The choice of embedding will, in general, affect the estimated divergence. This can be mitigated by calculating the divergence for many different embeddings and using the worst-case (i.e. largest) value. However, we found the impact to be relatively minor in practice, see Section-6.3, and SentenceBERT (15) to work well.



(a) Medal dataset (b) Political dataset (c) Amazon dataset (d) Reddit dataset

Fig. 4: Measured ϵ between redacted sensitive and safe datasets vs redaction level; random redaction. A lower value indicates better privacy. Also shown is the measured accuracy of a classification attack that tries to label which dataset the redacted sensitive text originated from (lower accuracy therefore equals greater privacy, with a classification accuracy of 50% corresponding to a random classifier).

4.2 Calculating (ϵ,δ)

To calculate ξ and ρ in equation (2), we first calculate D_α for a range of α values[4]. We then find a line that lies above the D_α vs α curve and select ρ as the slope of the line and ξ the intercept. Of course, many lines lie above the D_α curve, so we try to select one such that ρ and ξ are minimised. See for example Figure-3, which shows D_α vs α for the Medal medical dataset (the blue curve). This curve is upper bounded by the red line. The values of ρ and ξ corresponding to this red line are plugged into equation (3 to obtain the corresponding (ϵ,δ) privacy values.

Note. We use $\delta = 0.00008$ in all of our experiments as it is encouraged to keep $\delta < \frac{1}{n^2}$ where n is the number of input points (1).

5 Experiments

5.1 Datasets

We evaluate performance using the following datasets, each of which we split into "sensitive" and "safe" datasets.

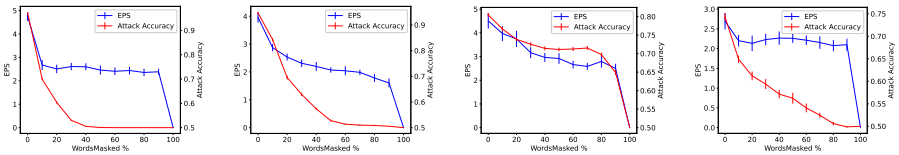
(i) Medal dataset (19)[5]. This dataset contains abstracts of medical papers, along with the diseases the abstract talks about. We partition this dataset into text with cancerous and non-cancerous diseases. Each dataset contains 2200 sentences. For our experiments, text with cancerous diseases was chosen to be the sensitive dataset.

(ii) Political dataset-(18)[6]. This contains comments on Facebook posts from 412 members of the United States Senate and House. Each comment is labeled

[4] We select the range to be large enough that D_α no longer increases as we increase α.

[5] https://huggingface.co/datasets/medal

[6] Data can be downloaded by following the instructions in the repository https://github.com/xuqiongkai/PATR



(a) Medal dataset (b) Political dataset (c) Amazon dataset (d) Reddit dataset

Fig. 5: Measured ϵ between redacted sensitive and safe datasets vs redaction level; more efficient redaction strategy. A lower value indicates better privacy. Also shown is the measured accuracy of a classification attack that tries to label which dataset the redacted sensitive text originated from (lower accuracy therefore equals greater privacy, with a classification accuracy of 50% corresponding to a random classifier).

with the corresponding Congressperson's party affiliation i.e. S ϵ {democratic, republican } We partition the dataset into text from users with Republican and Democrat political preferences. Each dataset contains 2000 sentences. For our experiments, text from users with Republican political preferences is chosen to be the sensitive dataset.

(iii) Amazon dataset[7]. This dataset contains product reviews from Amazon customers. We selected the reviews which were categorised as "drug-store" and "kitchen-appliances". For our experiments, the dataset with drug-store reviews was chosen to be the sensitive dataset.

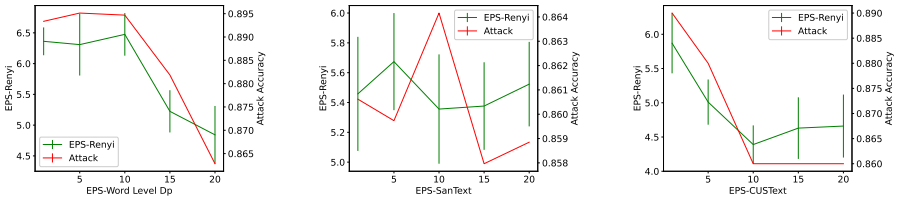
(iv) Reddit dataset[8]. This dataset contains post content from the subreddits r/depression and r/SuicideWatch. We partition this data into posts related to suicide and depression. Each dataset contains 2000 sentences. For our experiments, the text from the suicide subreddit was chosen to be the sensitive dataset.

5.2 Enhancing Privacy By Redaction

For each of the datasets we measured the Renyi-divergence as the percentage of words redacted using a random redaction policy was varied from 0 to 100%. The divergence values were then converted to ϵ values as explained previously. The results are shown in Figure 4.

To help gain confidence that redaction really is improving privacy, we carry out a simple classification attack. The redacted datasets are split into training and test data (90:10 split). A classifier is trained on this data, taking a sequence of words as input and outputting an estimate of whether the sentence came from the safe or sensitive datasets. Since there are only two classes and the data is balanced, a classification accuracy of 50% corresponds to a random classifier.

[7] https://huggingface.co/datasets/amazon_reviews_multi
[8] https://www.kaggle.com/general/256134



(a) Word-level DP (b) San-Text (c) CusText

Fig. 6: Measured ϵ-renyi and attack accuracy for various word-level DP approaches applied to Medal Dataset.

Figure 4 shows how the measured accuracy of this classifier varies with the redaction level for each of the datasets. It can be seen that the accuracy decreases as the redaction level increases, and that this decrease is roughly proportional to the decrease in the measured ϵ value.

5.3 Smarter Redaction

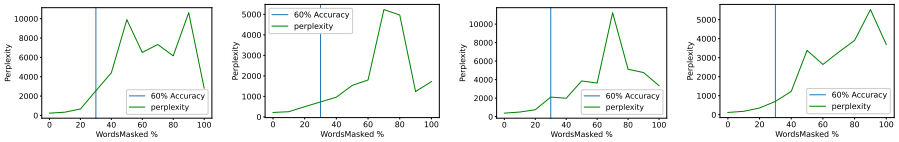
It can be seen from Figure 4 that when random redaction is used then relatively high levels of redaction are needed to ensure smaller ϵ values. Of course random redaction is rather crude, and smarter redaction approaches (in the sense that they achieve a target ϵ with fewer words redacted) are certainly possible.

We illustrate the scope for smarter redaction via a simple approach based on logistic regression weights. Namely, we took the logistic regression classifier from Section 5.2 and ranked the words from the datasets by the magnitude of the weight assigned to them by this classifier. We then redact the top p percent of these words from the datasets when redacting at level p.

Figure 5 plots the measured ϵ between the sensitive and safe datasets as the redaction level is increased in this way. It can be seen that a much lower level of redaction is now needed, compared to Figure 4, to obtain a given ϵ value. Also shown in Figure 5 is the measured accuracy of the simple classification attack as the redaction level is varied and it can be seen that the accuracy also now falls much more rapidly. For example, in Figure 4(a) a redaction level of around 90% is needed to reduce the accuracy of the attack to 60% (recall an accuracy of 50% corresponds to a random classifier, so 60% represents a high level of privacy), for the Medal dataset while with the smarter redaction strategy a redaction level of around 30% is sufficient to achieve this. Observe also that there is now a clear "knee" in the measured divergence vs redaction level curve, with the knee corresponding a low attack accuracy. It can be seen from Figure 5 that the behaviour is also similar for the other datasets studied.

5.4 Word-level DP

Word-level DP approaches sanitize text by converting each individual word to a vector embedding, adding noise to the embedding, and then mapping the noisy



(a) Medal dataset (b) Political Dataset (c) Amazon Dataset (d) Reddit dataset

Fig. 7: Measuring impact of privacy on utility. Next word prediction performance for LSTM trained on redacted dataset. Performance is measured on non-redacted data. The vertical line indicates the redaction level that reduces classification attack accuracy to 60%, taken from Figure 5.

embedding back to a word (8; 20; 5). In this section, we compare our approach to word-level DP approaches.

As discussed previously, word-level DP approaches aim to hide the information revealed by the individual words and so can fail to hide information revealed by the sentence as a whole[9].

We illustrate this by conducting the same attack as before on the word-level DP sanitized data, while also checking the ϵ (indicated by ϵ-renyi) between the sensitive and safe datasets. A high attack accuracy indicates that sensitive information is leaked from the sanitized sentences. Similarly, a high ϵ-renyi indicates that there are significant differences between sensitive and non-sensitive datasets.

Figure 8 shows the measured ϵ-renyi for the Medal dataset as the level of noise is increased (indicated by ϵ) for various word-level DP approaches. Also shown is the measured accuracy of our classification attack. It can be seen that even when a great deal of noise is added (low ϵ values), both the ϵ-renyi values and the attack accuracy remain high. Results for other datasets show similar behaviour and are provided in the Appendix [10].

5.5 Utility vs Privacy

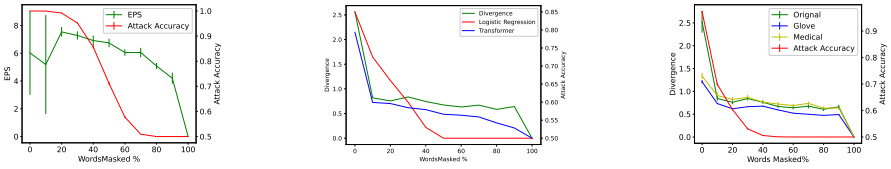
In general, we expect there to be a trade-off between privacy and utility. By redacting text to make a sensitive training dataset more private, the value of the sensitive training data is likely to be reduced because (i) redaction reduces the textual information contained in the sensitive dataset and (ii) by making the sensitive dataset more similar to an existing safe public dataset the added value over only using the public data for training is reduced.

To investigate this trade-off we trained a next-word-prediction model using PyTorch. A standard LSTM-RNN model[11] was used with two layers, each layer

[9] In particular, the DP analysis ignores correlations between the words in a sentence and so may greatly underestimate the information release. The impact of correlations on DP is well known and was first noted by (9).

[10] https://anonymous.4open.science/r/appendix_repo-F4CC

[11] Training code can be found at: https://github.com/pytorch/examples/tree/main/word_language_model



(a) (b) (c)

Fig. 8: 8a Measured ϵ and attack accuracy for cancer sentences when compared against IMDB reviews.8b Measured Renyi-divergence ($\alpha = 2$) and attack accuracy for logistic regression and BERT transformer classification attacks as the redaction level is increased. Medal dataset. 8c Measured Renyi-divergence ($\alpha = 2$) with different embeddings: (i) general-purpose sentenceBERT, (ii) fine-tuned medical sentenceBERT, (ii) Glove. Medal dataset.

having 200 hidden states and an input Embedding layer. The model has 4,041,675 parameters. A dictionary of all words was created from the dataset and input text was vectorized by replacing each word with its corresponding index in the dictionary. The model was trained using a negative log-likelihood loss.

The sensitive dataset was divided into a train-validation (90:10) split. The training data was redacted while validation data was not redacted. The model was then trained on the redacted training data and was tested against the held-out validation data. The model performance was evaluated by the measured perplexity (ppl) on the validation set, the lower the value of perplexity the better the model is at predicting a given sequence.

Figure 7 shows the measured perplexity on the validation data of the next word prediction model for each of the datasets studied as the redaction level is varied (for the smarter redaction approach of Section 5.3). The vertical line in the plot indicates the point where the measured attack classification accuracy is 60% (taken from Figure 5). It can be seen as expected, the perplexity increases as the redaction level increases. It can be seen that the perplexity increases with the level of redaction, as expected. However, for redaction levels up to 20-30% the perplexity of the model trained on the redacted dataset remains fairly close to the perplexity of the model trained on non-redacted dataset. A redaction level of 30% is sufficient to bring the attack classification accuracy down to 60% i.e a good degree of privacy can be obtained while preserving the utility of the dataset for model training.

6 Discussion

6.1 Choice of the Safe dataset

When the safe dataset is similar to the sensitive dataset, then we can expect that only a small amount of redaction is needed to bring the two datasets closer

together. Conversely, we expect that a higher level of redaction is needed to make very disparate datasets similar.

This is illustrated in Figure-8a, which can be directly compared with Figure-5a. In both cases the Medal cancer text is the sensitive dataset but in Figure-8a the safe dataset is IMDB review text while in Figure-5a it is Medal non-cancer text. It can be seen that with the IMDB data almost 80% of the words need to be redacted to get an attack accuracy close to 50% whereas with the Medal non-cancer text a redaction level of around 50% is sufficient.

The choice of safe dataset is therefore a privacy design parameter that can be used to manage the trade-off between privacy and utility in a fairly transparent manner.

6.2 More Powerful Attacks

It is important to stress that it is the Renyi-divergence that provides a sound measure of privacy, not the accuracy of any specific attack. In the previous sections we use a classification attack based on logistic regression to roughly verify privacy. However, this attack on its own does not provide any privacy guarantee.

For example Figure-8b shows the attack accuracy and Renyi-divergence as the level of redaction is varied. It can be seen that at redaction levels around 50-80% the attack accuracy is low (close to 50%, the accuracy of a random classifier). However, the Renyi-divergence remains relatively high at around 2.5 at these redaction levels, indicating that the datasets remain dissimilar. We therefore trained a more powerful BERT transformer model and used it to carry out the classification attack. Figure-8b, shows the measured attack accuracy. It can be seen that at 50-80% redaction the attack accuracy now remains relatively high, demonstrating the predictive power of the Renyi-divergence approach.

6.3 Choice of Embedding

The estimator in Section 4 maps text to a vector embedding and then estimates the Renyi-divergence between sets of vectors. It therefore depends on the choice of embedding used. It is problematic for a privacy approach to depend on the choice of embedding since (i) the properties of these embeddings remain poorly understood and (ii) an attacker can easily use a different embedding. For example, if a general purpose embedding is used in the Renyi-divergence but the attacker uses a domain specific embedding, a natural concern is that attacker may be able to extract information even when the Renyi-divergence estimate is low.

One of the great advantages of redaction is that it does not depend on the choice of embedding, but rather works directly with the text data[12]. We can then

[12] And one of the major deficiencies of all approaches tied to a single up front choice of embedding, such as word-level DP approaches.

calculate Renyi-divergence estimates for different choices of embeddings and use the largest value to evaluate privacy.

For example Figure-8c shows the measured Renyi-divergence estimates for three different choices of embedding: (i) a general-purpose pre-trained sentence-BERT embedding [13], (ii) sentenceBERT after fine-tuning on medical data and (iii) Glove[14] (14). sentenceBERT is a state of the art transformer embedding, Glove is an older embedding commonly used on word-level DP.

It can be seen from Figure-8c that the Renyi-divergence estimates for the two sentenceBERT embeddings are almost the same and consistently higher than the Renyi-divergence estimate using Glove i.e. Glove overestimates privacy. The consistency in the divergence estimates between the general purpose and fine-tuned sentenceBERT embedding indicates the robustness of the general purpose sentenceBERT and is why we use it in our earlier plots.

6.4 Limitations

Renyi-divergence is an estimate. Probably the main limitation of our approach is that it uses an estimate of the Renyi-divergence rather than the true value. We partially mitigate this by also estimating confidence intervals. However use of an estimate seems unavoidable since the true divergence cannot be calculated for realistic text data, and for similar reasons theoretical differential privacy guarantees are intractable. We argue that adopting a pragmatic approach and using estimates provides a way forward that is both useful and represents significant progress over the state of the art in text privacy. This is particularly pressing given the prevalence of text data and the current great interest in using it to train large language models.

7 Conclusions

We revisit the question of how to sanitise sensitive text data so that it can be used for model training while preserving privacy. The great majority of the existing literature on privacy enhanced model training gains privacy by adding noise to gradients used for training. The amount of noise added needs to scale with the number of model parameters since the DP sensitivity scales with this. When the number of model parameters is large (as it usually is for language models), the amount of noise needed is considerable and adversely affects model utility. Sampling of the input data can be used to boost privacy, but requires effective anonymisation which can be hard to achieve in practice and reduces the volume of training data available. The data sanitisation approach that we consider complements this line of work and offers new ways to manage the trade-off between privacy and utlity.

[13] https://www.sbert.net/

[14] The embedding vector of each word in a sentence is calculated, and the mean of these vectors is used as the sentence embedding.

Bibliography

[1] Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. p. 308–318. CCS '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978318, https://doi.org/10.1145/2976749.2978318

[2] Bosch, N., Crues, R., Shaik, N., Paquette, L.: " hello,[redacted]": Protecting student privacy in analyses of online discussion forums. Grantee Submission (2020)

[3] Brown, H., Lee, K., Mireshghallah, F., Shokri, R., Tramèr, F.: What does it mean for a language model to preserve privacy? In: 2022 ACM Conference on Fairness, Accountability, and Transparency. p. 2280–2292. FAccT '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3531146.3534642, https://doi.org/10.1145/3531146.3534642

[4] Bun, M., Steinke, T.: Concentrated differential privacy: Simplifications, extensions, and lower bounds (2016)

[5] Chen, S., Mo, F., Wang, Y., Chen, C., Nie, J.Y., Wang, C., Cui, J.: A customized text sanitization mechanism with differential privacy. In: Findings of the Association for Computational Linguistics: ACL 2023. pp. 5747–5758. Association for Computational Linguistics, Toronto, Canada (Jul 2023). https://doi.org/10.18653/v1/2023.findings-acl.355, https://aclanthology.org/2023.findings-acl.355

[6] Doudalis, S., Kotsogiannis, I., Haney, S., Machanavajjhala, A., Mehrotra, S.: One-sided differential privacy (2017)

[7] Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography. pp. 265–284. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[8] Feyisetan, O., Balle, B., Drake, T., Diethe, T.: Privacy- and utility-preserving textual analysis via calibrated multivariate perturbations. In: Proceedings of the 13th International Conference on Web Search and Data Mining. p. 178–186. WSDM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3336191.3371856, https://doi.org/10.1145/3336191.3371856

[9] Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. p. 193–204. SIGMOD '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/1989323.1989345, https://doi.org/10.1145/1989323.1989345

[10] Mattern, J., Weggenmann, B., Kerschbaum, F.: The limits of word level differential privacy. In: Findings of the Association

for Computational Linguistics: NAACL 2022. pp. 867–881. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.findings-naacl.65, https://aclanthology.org/2022.findings-naacl.65

[11] Mironov, I.: Rényi differential privacy. In: 2017 IEEE 30th Computer Security Foundations Symposium (CSF). IEEE (Aug 2017). https://doi.org/10.1109/csf.2017.11, http://dx.doi.org/10.1109/CSF.2017.11

[12] Murugadoss, K., Rajasekharan, A., Malin, B., Agarwal, V., Bade, S., Anderson, J.R., Ross, J.L., William A. Faubion, J., Halamka, J.D., Soundararajan, V., Ardhanari, S.: Building a best-in-class automated de-identification tool for electronic health records through ensemble learning. medRxiv (2021). https://doi.org/10.1101/2020.12.22.20248270, https://www.medrxiv.org/content/early/2021/02/23/2020.12.22.20248270

[13] Noshad, M., Moon, K.R., Sekeh, S.Y., Hero, A.O.: Direct estimation of information divergence using nearest neighbor ratios. In: 2017 IEEE International Symposium on Information Theory (ISIT). IEEE (Jun 2017). https://doi.org/10.1109/isit.2017.8006659, http://dx.doi.org/10.1109/ISIT.2017.8006659

[14] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014), http://www.aclweb.org/anthology/D14-1162

[15] Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks (2019)

[16] Shi, W., Shea, R., Chen, S., Zhang, C., Jia, R., Yu, Z.: Just fine-tune twice: Selective differential privacy for large language models. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 6327–6340. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Dec 2022), https://aclanthology.org/2022.emnlp-main.425

[17] Shokri, R., Shmatikov, V.: Privacy-preserving deep learning. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. p. 1310–1321. CCS '15, Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2810103.2813687, https://doi.org/10.1145/2810103.2813687

[18] Voigt, R., Jurgens, D., Prabhakaran, V., Jurafsky, D., Tsvetkov, Y.: RtGender: A corpus for studying differential responses to gender. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (May 2018), https://aclanthology.org/L18-1445

[19] Wen, Z., Lu, X.H., Reddy, S.: MeDAL: Medical abbreviation disambiguation dataset for natural language understanding pretraining. In: Proceedings of the 3rd Clinical Natural Language Processing Workshop. Association for Computational Lin-

guistics (2020). https://doi.org/10.18653/v1/2020.clinicalnlp-1.15, https://doi.org/10.18653%2Fv1%2F2020.clinicalnlp-1.15

[20] Yue, X., Du, M., Wang, T., Li, Y., Sun, H., Chow, S.S.M.: Differential privacy for text analytics via natural text sanitization. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 3853–3866. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.findings-acl.337, https://aclanthology.org/2021.findings-acl.337

[21] Zhao, X., Li, L., Wang, Y.X.: Provably confidential language modelling. In: Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 943–955. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.naacl-main.69, https://aclanthology.org/2022.naacl-main.69

Exploring Distribution Learning of Synthetic Data Generators for Manifolds

Sonakshi Garg[1] [0000−0002−7204−8228] and Vicenç Torra[1] [0000−0002−0368−8037]

Umeå University, Umeå, Sweden {sgarg,vtorra}@cs.umu.se

Abstract. In the era of data protection regulations like GDPR, safeguarding sensitive information has become paramount, prompting the exploration of synthetic data generation as a privacy-preserving alternative. Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE), among other tools, have become popular for synthetic data generation. Despite their effectiveness, these models often carry the perception of being black boxes due to their complex learning mechanisms. Understanding the intricate behaviors of data within GAN or VAE poses a significant challenge, particularly with high-dimensional datasets. This is essential from privacy perspective as one can use synthetic data instead of original data and this can be considered as an alternative to anonymization. Our study aims to assess the distribution learning capabilities of synthetic data generators. Our methodology centers on artificially created datasets, such as swish roll and S-curve distributions, which offer easy visualization in R^n space. Additionally, we evaluate point datasets containing discontinuous points to determine whether GAN and VAE comprehend the discontinuity behavior of datasets. By evaluating the data processed by GAN and VAE, we aim to reveal their learning capabilities and disentangle the complexities of synthetic data generation. Our research shifts the focus from real-world image datasets to artificially generated datasets, enabling exploration of commonly encountered distributions in low-dimensional spaces. Despite widespread recognition of GAN in image synthesis, achieving satisfactory results often requires employing numerous tricks due to training instability. We found that VAE exhibit a superior understanding of the underlying distribution of points in R^n space compared to GAN. This inclination towards VAE arises from their more stable training process, inherent ability to capture latent structures within the data, and faster convergence compared to GAN.

Keywords: Manifold Learning · Privacy · Synthetic Data Generators · Generative Adversarial Networks · Variational Auto Encoder

1 Introduction

In today's data-driven world, privacy concerns, especially under GDPR [6], highlight the need for effective data protection. Two key approaches are data anonymization [19,4] and synthetic data generation. While data anonymization has been extensively studied, synthetic data generation is a growing field. Generative Adversarial Networks (GANs) [7] and Variational AutoEncoders (VAEs) [10] are prominent methods for this. GANs use adversarial training with a generator and discriminator to create realistic data, whereas

VAEs use probabilistic inference to encode and decode data. Both models excel but are often seen as black boxes with limited insight into their internal workings [18].

While GANs and VAEs excel in image synthesis, their black-box nature complicates visualization and understanding, particularly with complex datasets like ImageNet [3]. To address this, our methodology utilizes simpler, artificially created datasets like Swish Roll and S-Curve, which can be easily visualized in lower-dimensional spaces. This approach helps illuminate the learning behavior of these models and their synthetic data generation capabilities. While GANs perform well on some datasets [9], their effectiveness can vary with more complex data distributions [2,14], making simpler datasets ideal for exploring their strengths and limitations. Thus, our research shifts focus to artificially generated datasets, allowing us to explore the learnability of commonly encountered distributions in low-dimensional spaces. Through this endeavor, we seek to enhance understanding of synthetic data generation methods and their applicability across various domains.

We use UMAP [12] leveraging its unique capability of inverse transformation, which is not possible with many other approaches to transform datasets into a latent space, generate synthetic data with GANs and VAEs, and then reconstruct the data using UMAP's inverse transform. This approach allows us to evaluate the performance of both manifold learning and synthetic data generators. Our experiments, including on datasets like Swish Roll and S-Curve, show that VAEs better capture data distribution compared to GANs. We also found that GANs often struggle with instability and require complex optimization. Our research addresses the challenges of visualizing and evaluating synthetic data generation for manifold datasets, which are distinct from traditional image datasets.

The main contributions of the paper are as follows.

1. We explore how synthetic data generators (GANs and VAEs) perform on artificially created datasets, focusing on privacy concerns.
2. Our methodology involves using UMAP to map datasets to a latent space for synthetic data generation and subsequent reconstruction to assess learning capabilities.
3. We evaluate these models on various datasets, including Swish Roll, S-Curve, and image datasets, uncovering challenges and insights.
4. Our findings show that VAEs outperform GANs in capturing data distribution, and we address the difficulties encountered in GAN training.
5. We address challenges in synthetic data generation for high-dimensional manifold datasets, aiming to improve understanding and performance.

2 Preliminaries

In this section we will explain the important concepts that are used in this paper.

2.1 Manifold Learning

Mathematically, a manifold is a topological space locally resembling Euclidean space, meaning it can be approximated by Euclidean space in small neighborhoods. Formally, a manifold is a space where each point has a neighborhood that is homeomorphic to an open subset of n-dimensional Euclidean space.

Definition 1. *(Manifold Learning) Given a finite set of data points $x_1, ...x_n \in \mathbb{R}^D$ be in a D-dimensional space, Manifold learning aims to find the low-dimensional points $y_1, ...y_n \in \mathbb{R}^d$ where $d \ll D$ such that Euclidean relationship between (y_i, y_j) reflects the intrinsic non-linear relationships between (x_i, x_j) [5].*

In manifold learning, the hypothesis is that data points reside on a low-dimensional manifold $\mathbb{R}^d$, embedded within a higher-dimensional ambient space $\mathbb{R}^D$ [20]. The goal is to find a mapping from $\mathbb{R}^D$ to a lower-dimensional space while preserving geometric properties.

2.2 Synthetic Data Generators

Synthetic data generators create data that mimics original data to address privacy concerns and reduce reliance on actual data. Regulatory frameworks like GDPR highlight the importance of data protection. While one method involves anonymizing data, another involves generating synthetic data that replicates original data structures. GANs [7] have recently become popular for this purpose. GANs have been applied in fields like natural language processing and computer vision. The basic GAN architecture, known as Vanilla GAN, generates realistic data across various domains. For more control over output, Conditional GAN (cGAN) [13] allows for conditional image generation based on factors like class labels. Another specialized variant, Deep Convolutional GAN (DCGAN) [16], improves image synthesis by using convolutional layers in both the generator and discriminator, enhancing training stability and capturing spatial hierarchies in images. Conditional Tabular GAN (CTGAN) [21] is a robust GAN variant designed for diverse real-world datasets, capturing data heterogeneity effectively. In addition to GANs, Variational Autoencoders (VAEs) [10] are another prominent generative model family. VAEs are autoencoders that regularize the encoding distribution to ensure a well-structured latent space for generating new data. They are trained to minimize reconstruction error while avoiding overfitting.

3 Methodology

In this section we propose our methodology for visualizing the manifolds to determine whether the manifolds generated using synthetic data generators converge to real data manifolds. We aim to assess the ability of generators to understand and replicate the underlying distribution of the data it is trained on. We describe our methodology with step-by-step explanation as follows.

1. Dataset Selection: Start by selecting a real-world high-dimensional dataset with a manifold structure. For this task, we use the MNIST [3] dataset.

2. Train a Manifold Learning Model: Train Uniform Manifold Approximation and Projection (UMAP) [12] on the chosen dataset to transform it into a lower-dimensional space. UMAP is selected for its ability to preserve both local and global structures, functionality of inverse-transformation and its faster computation compared to t-SNE.

3. Reconstruction to Original Space: Use UMAP's inverse mapping to reconstruct the data back to its original high-dimensional space, ensuring the model handles manifold structures effectively.

4. Generation of Artificial Data: Test the trained model on artificial datasets in $\mathbb{R}^4$ and $\mathbb{R}^2$ to visualize and understand the dataset, the lower-dimensional transformations, and the synthetic data, which is not feasible with high-dimensional real-world data.

5. Test the Manifold Learning Model: Apply the trained manifold learning model to transform artificial datasets into the latent space for visualization. This step assesses the model's ability to preserve proximity between points from high-dimensional to lower-dimensional spaces and allows for performance evaluation of the manifold learning algorithm.

6. Synthetic Data Generators: Introduce generators like GAN and VAE to create synthetic data from the latent space. This step evaluates how well the generated data resembles real data and ensures privacy by retaining data structure while introducing variability to avoid exact replication.

7. Synthetic Data Reconstruction: Use the inverse transformation function to reconstruct synthetic data back to its original dimensional space. Compare the reconstructed data with the original synthetic data to evaluate the accuracy of the reconstruction process.

4 Experimental Setup

In this section we describe the setup of our experiments which includes description of the datasets and some details about the synthetic data generator architectures.

4.1 Datasets Description

We selected various datasets with manifold structures to evaluate our approach comprehensively. Manifold structure represents intricate data distribution patterns in a high-dimensional space that can't be fully captured by traditional linear methods. These include high-dimensional point datasets like Swish Roll [11] and S-curve residing in $\mathbb{R}^4$, and lower-dimensional datasets such as Concentric Circles, Mixture of Gaussian Points, and Two-Half Circles residing in $\mathbb{R}^2$ plane, all generated using the sklearn library [15]. Each dataset consists of 4000 samples with each sample representing a fixed point in $\mathbb{R}^n$. Additionally, we used the MNIST dataset to test manifold learning on real-world complex data distributions. This variety allows us to assess the robustness and effectiveness of our manifold learning techniques and synthetic data generators.

4.2 Architectures

We trained our model using Vanilla GAN, where both the generator and discriminator networks were built. The generator had four dense layers with Leaky ReLU activations and a tanh activation at the output. The discriminator mirrored this setup but used a sigmoid activation at the output. After updating the discriminator, it was frozen, and the generator was trained on fake data, with loss backpropagated to adjust its weights. Next, we used the DCGAN architecture for image synthesis. The DCGAN generator includes dense layers, Batch Normalization, and a Convolution1D layer for upsampling. Unlike traditional GANs, the DCGAN discriminator handles image inputs rather than vectors,

and both generated and original images are evaluated by it. Leaky ReLU activations are used in this setup, while the discriminator's operation follows the traditional GAN process.

Alongside Vanilla GAN and DCGAN, we used Conditional Tabular GAN (CTGAN) and Differentially Private GAN (DPGAN) for synthetic data generation. CTGAN was used for its specialization in tabular data, while DPGAN's architecture was employed to ensure differential privacy. This approach allowed us to assess the effectiveness of these GAN variants in generating synthetic data and addressing privacy concerns across different domains. We used VAE for synthetic data generation, featuring an encoder with two dense ReLU layers to produce the mean and log variance of the latent distribution, and a decoder with two dense ReLU layers and a sigmoid output for reconstruction. The sampling layer utilizes the reparameterization trick to generate latent vectors. All models, including VAE, were trained with the Adam optimizer at a learning rate of 10^{-4}.

5 Results and Discussion

In this section, we delve into the outcomes of our experiments and provide comprehensive discussion on them.

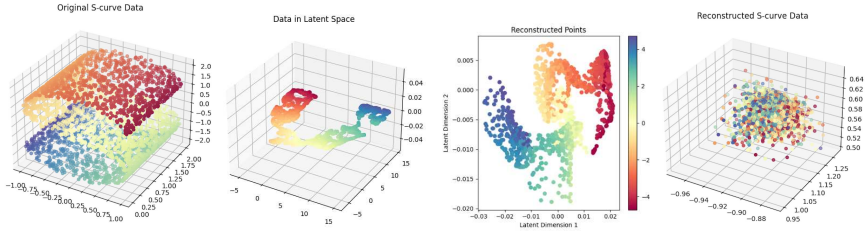
5.1 Visualizing Synthetic Generation from S-Curve Transformation

Figure 1 outlines our methodology applied to the S-Curve dataset, beginning with its creation using the 'make-s-curve' function from sklearn. We first transformed this dataset into a 2D plane using a pre-trained UMAP model, originally trained on the MNIST dataset (Figure 1b). This transformation effectively retained the original shape and geometry of the S-Curve, validating the manifold hypothesis which posits that proximity in high-dimensional space is preserved in lower-dimensional representations. The labeled points in the 2D plane confirm the accurate univariate positioning based on the primary dimensions of the manifold.

Next, we generated synthetic data using a Variational Autoencoder (VAE) trained on the UMAP-transformed data (Figure 1c). VAEs, known for capturing the underlying distribution of input data, utilized the lower-dimensional structure provided by UMAP to produce synthetic data that mirrors the original dataset's patterns. The latent space representation by the VAE further confirmed its efficacy in capturing the essential features of the S-Curve. Finally, we used the inverse transform functions of UMAP to reconstruct the data in the original space (Figure 1d). While the reconstructed points generally clustered around the central region, the dispersion was limited, highlighting the challenge of precisely recovering the high-dimensional structure and indicating areas for improvement in the transformation process.

5.2 Unrolling the Swish Roll: Exploring Manifold Transformation

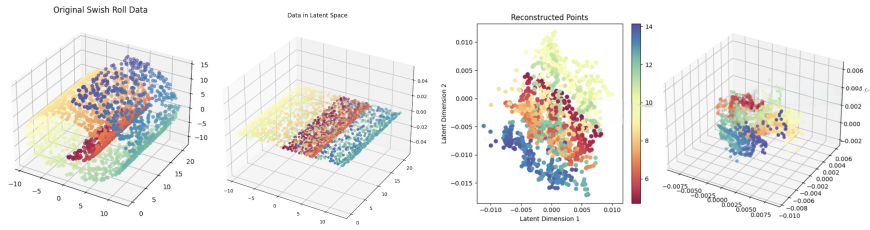
Figure 2 shows the Swish roll dataset, initially visualized in Figure 2(a). Transforming this data into 2D using UMAP (Figure 2b) reveals its intricate structure and relation-



(a) Original Data (b) Data Transform (c) Synthetic Data (d) Reconstructed data

Fig. 1: S-Curve Dataset

ships, illustrating manifold learning principles. UMAP effectively captures and preserves the dataset's complex high-dimensional patterns in a lower-dimensional space, highlighting its utility for creating meaningful data representations. In Figure 2(c), synthetic data generated by VAE from UMAP-transformed points displays distinct labels and closely mimics the original data, with minor dissimilarities indicating noise for privacy. This highlights VAE's effectiveness in capturing the original data's characteristics while preserving privacy. Figure 2(d) shows the reconstruction of original data using the manifold model's inverse transform. Although points cluster by label, overlapping regions complicate accurate reconstruction, reflecting challenges in handling high-dimensional data with noise and overlapping surfaces.



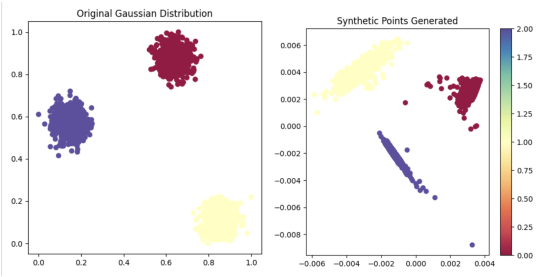
(a) Original Data (b) Data Transform (c) Synthetic Data (d) Reconstructed data

Fig. 2: Swish Roll Dataset

5.3 Understanding 2D Point Datasets

In our analysis of synthetic data generators, we applied our methodology to 2D point datasets. Figure 3(a) shows Gaussian clusters generated using sklearn's make-blobs function, with three clusters having a standard deviation of 0.2. We then used a VAE to replicate this data, as seen in Figure 3(b). The VAE-generated points form three distinct clusters, though some points are stretched within each cluster. This indicates the VAE's success in producing data similar to the original while preserving privacy through slight

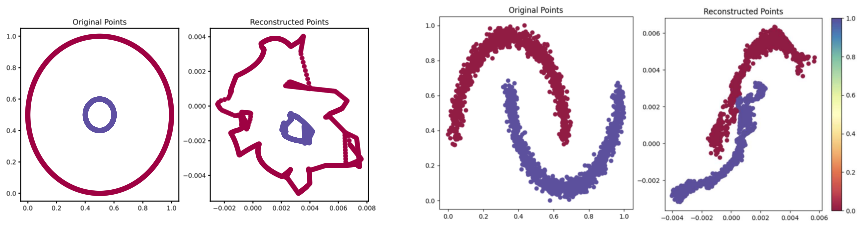
dissimilarities. The VAE effectively learns and reproduces the discontinuous nature of the original data points, despite not following a continuous pattern. This contrasts with findings in [17], where the model struggled with data discontinuity due to assumptions of a continuous latent space.



(a) Original Data (b) Synthetic Data

Fig. 3: Mix of Gaussian Points

In Figure 4, we observe a similar pattern with the concentric circles dataset and the two half circles dataset. In both visualizations, the original datasets exhibit a discontinuous nature with distinct geometrical patterns. This process ensures that the privacy of the original data's structure and geometry is maintained while generating synthetic data. The produced synthetic data closely resembles the structure and geometry of the original datasets while incorporating some variation. This behavior arises because the VAE learns the distribution of the data and generates new points based on this learned distribution, thereby preserving the essential characteristics of the original data while introducing slight deviations.

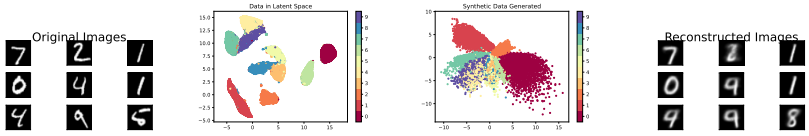


(a) Concentric Circles (b) Two Half Circles (c) Synthetic Data

Fig. 4: Concentric and Two Half Circles

5.4 Visualizing Real-World Data

We explored the capabilities of manifold learning and synthetic data generators using the MNIST dataset, a widely-used real-world dataset in machine learning. Firstly, we visualized the original MNIST dataset, consisting of 70,000 handwritten digit images sized at 28x28 pixels, as shown in Figure 5(a). Next, we transformed this data into a 2D latent space using manifold learning, resulting in well-separated clusters representing different digits, as depicted in Figure 5(b). We then generated synthetic data from this transformed data using a VAE, shown in Figure 5(c), where the data points appear clustered closely together. The reconstructed data, seen in Figure 5(d), closely resembles the original data. However, when dealing with real-world image datasets like MNIST, we can only visualize the original and reconstructed data. Once the data is transformed into a latent space, visualization is limited to observing if the data points are well-separated according to their labels. This black-box-like mechanism of synthetic data generators makes it difficult to visualize how the geometry of data points changes at each step. This is why we used artificially created datasets in 4D and 2D planes, allowing us to visualize the workings of synthetic data generators more effectively.



(a) Original Data (b) Data transformed (c) Synthetic Data (d) Reconstructed Data

Fig. 5: MNIST Dataset

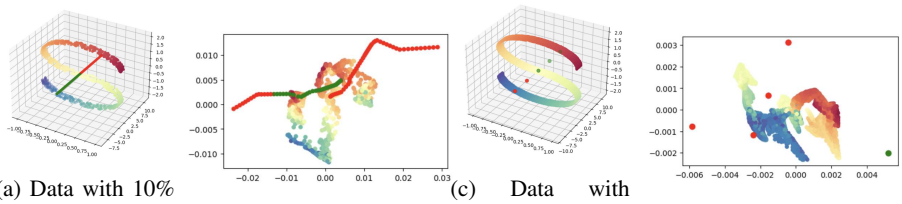
5.5 Privacy Risk Assessment in VAE

We extended our privacy analysis of VAEs by adding new artificial points to the original S-Curve dataset and examining the VAE's ability to regenerate these points. Figure 6(a) shows the original data with 10% additional points arranged along a straight line, marked in red and green. Figure 6(b) displays the synthetic data generated by the VAE, which accurately regenerates the added points. This demonstrates the VAE's proficiency in learning and generalizing from the dataset, effectively capturing the underlying structure, including the newly introduced points. When the added points are numerous and systematically distributed, as in Figure 6(a), the model's ability to regenerate them in Figure 6(b) indicates that it has effectively learned the underlying structure of the data, including the newly introduced points. This capability highlights the VAE's proficiency in learning and generalizing from the dataset, ensuring that it can produce realistic synthetic data while preserving the distribution of added data points.

If the VAE regenerates very few newly added points precisely at their original locations, it indicates a potential privacy leak due to memorization of individual samples

rather than learning general patterns. Figures 6(c) and 6(d) illustrate this scenario. In Figure 6(c), only 0.01% of points are newly added and strategically placed. In Figure 6(d), the synthetic data shows these points in different locations, indicating the VAE did not memorize the specific data samples but learned general patterns instead. The scattered positions of the newly added points demonstrate that the VAE preserves the overall S-curve structure without retaining exact data points, reducing the risk of privacy leakage. In Figure 6(c), the three newly added points (two green and one red) coincide with the S-Curve data points but are not regenerated at the same positions by the VAE, indicating they were not memorized. Instead, these points are scattered across the S-Curve structure. Figure 6(d) shows that while the VAE preserves the overall S-shape, it does not retain the exact positions of individual data points, as evidenced by the color spectrum. This observation is crucial for privacy, demonstrating that the VAE does not memorize specific, potentially sensitive information, thereby reducing the risk of privacy leakage.

This visualization assesses privacy risks in VAE data generation. Accurate regeneration of specific, newly added points by the VAE suggests memorization of exact details rather than learning general patterns, potentially leading to privacy breaches. Ensuring the VAE does not memorize individual data points is crucial for maintaining privacy and avoiding the revelation of sensitive information.



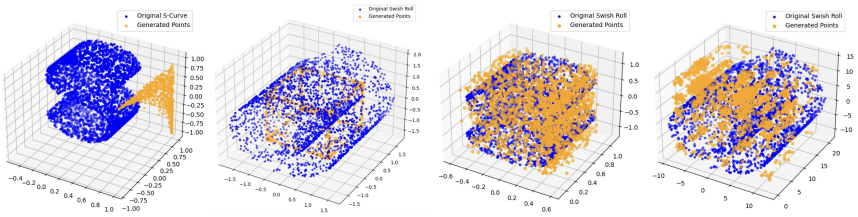
(a) Data with 10% new points (b) Regenerated point (c) Data with 0.01% new points (d) Regenerated point

Fig. 6: Privacy Risk Assessment in VAE

5.6 Visualization with Diverse GAN Architectures

We tested various GANs for synthetic data generation, including Vanilla GAN, DCGAN, CTGAN, and DPGAN. Starting with Vanilla GAN on the S-Curve dataset, as shown in Figure 7(a), we found that it performed poorly, with generated data being far from the original points. This indicates that Vanilla GAN struggled with the 3D point data's intrinsic dimensionality. We tested various GANs on all datasets, but obtained similar results across them. For simplicity, we are only presenting the findings for the S-Curve dataset. We applied DCGAN to the Swish Roll dataset (Figure 7(b)), finding that while the generated points overlapped with the original, they were disorganized. Using CTGAN on the S-Curve dataset (Figure 7(c)), the generated points also showed significant overlap with the original data, and the latent space representation was scattered, highlighting CTGAN's difficulty in capturing the 3D geometry.

We tested DPGAN on the Swish Roll dataset (Figure 7(d)), but found that the generated points were scattered and lacked geometric fidelity. GANs, known for their instability and slow convergence [7,16] compared to VAEs [1], struggled with manifold data. This led us to favor VAE, which provided more reliable results. Unlike GANs, which often face mode collapse, VAE effectively captured the diverse structures in our datasets [8], making it the preferred choice for our synthetic data generation. This discrepancy sheds light on why GAN struggled with our manifold data, which consists of data points in $\mathbb{R}^n$ space rather than images.



(a) Vanilla GAN (b) DCGAN (c) CTGAN (d) DPGAN

Fig. 7: Results with other types of GAN

6 Conclusion and Future Works

In this study, we examined the effectiveness of synthetic data generators, GAN and VAE, for privacy-preserving data. Given their black-box nature and challenges with high-dimensional data, we used artificially created datasets in $\mathbb{R}^n$ and applied UMAP for data transformation. This approach allows us to visualize and evaluate the performance of GAN and VAE throughout their training, providing clearer insights into their learning capabilities. Our findings indicate that VAE demonstrate a superior ability to understand and learn the intrinsic structure of our artificial point dataset compared to GAN. Furthermore, when attempting to reconstruct the original data using inverse transformation, we observed interesting outcomes. For instance, in the case of the swish roll dataset, the data points were clustered according to their labels, while in the s-curve dataset, the produced samples were situated between the clusters of data. This outcome leads to generalization of data which is promising from a privacy perspective, as the generated data samples fall within the range of the original data points without fully revealing their exact shape and structure. However, the effectiveness of this process depends on the compression and inverse transformation techniques employed. Nevertheless, privacy always comes at the cost of utility. Future research could focus on developing techniques that offer both efficient transformation and inverse transformation with minimal loss. Additionally, there is a need for synthetic data generators tailored specifically for point datasets in $\mathbb{R}^n$ rather than images, which could further enhance privacy-preserving data generation techniques.

References

1. Bond-Taylor, S., Leach, A., Long, Y., Willcocks, C.G.: Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. IEEE transactions on pattern analysis and machine intelligence **44**(11), 7327–7347 (2021)
2. Brock, A., Donahue, J., Simonyan, K.: Large scale gan training for high fidelity natural image synthesis. arXiv preprint arXiv:1809.11096 (2018)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
4. Dwork, C.: Differential privacy. In: International colloquium on automata, languages, and programming. pp. 1–12. Springer (2006)
5. Garg, S., Torra, V.: K-anonymous privacy preserving manifold learning. In: the 20th International Conference on Security and Cryptography, Rome, Italy, July 10-12, 2023. vol. 1, pp. 37–48 (2023)
6. GDPR:EU: Gdpr compliance. https://gdpr.eu/ (2020)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Communications of the ACM **63**(11), 139–144 (2020)
8. Huang, H., Yu, P.S., Wang, C.: An introduction to image synthesis with generative adversarial nets. arXiv preprint arXiv:1803.04469 (2018)
9. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2017)
10. Kingma, D.P., Welling, M., et al.: An introduction to variational autoencoders. Foundations and Trends® in Machine Learning **12**(4), 307–392 (2019)
11. Marsland, S.: Machine learning: an algorithmic perspective. Chapman and Hall/CRC (2011)
12. McInnes, L., Healy, J., Melville, J.: Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426 (2018)
13. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014)
14. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier gans. In: International conference on machine learning. pp. 2642–2651. PMLR (2017)
15. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. the Journal of machine Learning research **12**, 2825–2830 (2011)
16. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
17. Rege, A., Monteleoni, C.: Evaluating the distribution learning capabilities of gans. arXiv preprint arXiv:1907.02662 (2019)
18. Savage, N.: Breaking into the black box of artificial intelligence (2022)
19. Sweeney, L.: k-anonymity: A model for protecting privacy. International journal of uncertainty, fuzziness and knowledge-based systems **10**(05), 557–570 (2002)
20. Tenenbaum, J.B., Silva, V.d., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. science **290**(5500), 2319–2323 (2000)
21. Xu, L., Skoularidou, M., Cuesta-Infante, A., Veeramachaneni, K.: Modeling tabular data using conditional gan. Advances in neural information processing systems **32** (2019)

DPM Session 2: Privacy & Applied Cryptography

HEDAS: Secure and Efficient Distributed OLAP using Fully Homomorphic Encryption

Yu Tian[1], Tianxiang Shen[1], Qi Hu[1], Wei Chen[1], Heming Cui[1,3], and Ji Qi[2(✉)]

[1] The University of Hong Kong, Hong Kong, China
{tianyuk,txshen2,h3009790,h3011068,heming}@cs.hku.hk
[2] Institute of Software, Chinese Academy of Sciences, Beijing, China
qiji@iscas.ac.cn
[3] Shanghai AI Laboratory, Shanghai, China

Abstract. The popularity of cloud computing has revolutionized Online Analytical Processing (OLAP), yet risks of privacy leakage limit the use of public clouds in security-critical scenarios, such as healthcare and finance. Fully Homomorphic Encryption (FHE) is a promising solution that enables computations on encrypted data without decryption. However, the performance overhead of FHE hinders its practical use in OLAP. Existing FHE-based OLAP systems primarily focus on single-machine optimization and lack proper co-design with OLAP characteristics, leading to suboptimal effectiveness. Our key idea is to distribute FHE-based OLAP across multiple machines, inspired by the MapReduce model, to reduce end-to-end latency.

In this paper, we propose HEDAS[4], the first distributed FHE-based OLAP system that achieves secure and efficient OLAP. HEDAS effectively addresses the limitations of FHE by introducing the *pre-group* operation and two key stages: *Secure Map (SMap)* and *Secure Reduce (SReduce)*. These stages securely and efficiently execute filtering and aggregation operations in OLAP. By leveraging the MapReduce model, HEDAS effectively distributes FHE-based OLAP across multiple machines, achieving a notable reduction (∼44.1%) in end-to-end latency using four computing nodes, with further reductions as the number of computing nodes increases. Additionally, two case studies are conducted to address the optimization of the filter bottleneck within HEDAS, and evaluation experiments demonstrate the efficiency and scalability of HEDAS compared to the state-of-the-art single-machine FHE-based OLAP system.

Keywords: Fully homomorphic encryption (FHE) · Online analytical processing (OLAP) · MapReduce.

1 Introduction

The widespread adoption of cloud computing has revolutionized database systems, including Online Analytical Processing (OLAP) [8], yet concerns about

✉ Ji Qi is the corresponding author.
[4] HEDAS stands for Homomorphic Encryption-based Distributed Analytical System.

privacy breaches have restricted the applicability of public clouds in scenarios where data confidentiality is of utmost importance [26,41,42]. While the public cloud can provide OLAP with scalability and flexibility advantages, industries such as healthcare and finance face the requirement to safeguard sensitive data of patients and customers in compliance with regulations and to maintain trust.

To ensure the security of data on public clouds, traditional encryption techniques (e.g., AES [38] and RSA [35]) are unsuitable for OLAP scenarios due to their inability to perform computations and analysis on encrypted data. Instead, Fully Homomorphic Encryption (FHE) [14] overcomes this limitation by enabling computations on encrypted data without the need for decryption, thereby allowing the utilization of cloud computing resources while preserving data confidentiality. As a result, there has been a surge in research efforts to design FHE-based OLAP systems [34,2].

However, although FHE is utilized in some OLAP systems to meet security requirements, it suffers from substantial performance overhead. For instance, when executing the CKKS algorithm [5] on CPU, it typically incurs a computational complexity overhead of $\sim 10^4$-$10^5\times$ compared to plaintext computations.

Much existing work has explored ways to accelerate FHE-based OLAP systems; however, they haven't fully satisfied the performance requirements for practical applications. There exists two major categories of work, one focusing on single-machine optimization by improving FHE algorithms [2,34,33,36], and the other utilizing heterogeneous computing acceleration units (e.g., GPU [40,43] and FPGA [27]) to enhance the performance of critical analytic operators under FHE. While these methods achieved notable performance improvements, they solely focus on the single-machine level and lack deep integration with the characteristics of OLAP. OLAP applications are commonly deployed in a distributed manner, such as data lakes [21] and data warehouses [1], providing an opportunity to leverage the distributed workload computation capabilities across multiple machines.

One strawman approach to scaling existing FHE-based OLAP systems in a distributed manner is by utilizing the classical MapReduce model [11], a well-established practice in distributed data processing, making it an ideal choice for our early exploration. As shown in the left part of Figure 1, the MapReduce model consists of a coordinator node and multiple worker nodes responsible for task assignment and data processing, respectively. MapReduce abstracts the data processing into two stages: Map and Reduce. During the Map stage, each worker node processes a portion of the data and generates intermediate key-value results, which involves filtering out unnecessary data that doesn't meet the specified conditions. Subsequently, the system groups the intermediate results with the same key to the corresponding Reduce node, known as the data shuffle. The Reduce nodes then aggregate the intermediate results and produce the final results. Compared to single-machine processing, MapReduce provides a scalable and fault-tolerant distributed abstraction, effectively utilizing the computing resources of multi-node clusters.

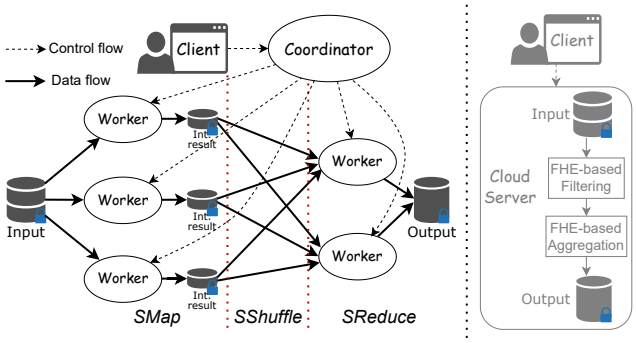


Fig. 1: An overview of the HEDAS model compared to single-machine processing.

However, directly extending FHE-based OLAP systems using the MapReduce model is non-trivial, as it introduces new challenges. Specifically, (1) the result of FHE comparison is encrypted randomly, making it impossible to group the intermediate results, rendering the data shuffle in the MapReduce model impractical. When performing OLAP on a single machine, there is no requirement to handle partial data, eliminating the necessity for data shuffle. On the contrary, in a distributed MapReduce model, it becomes essential to shuffle intermediate results after filtering; otherwise, subsequent aggregation operations cannot be performed. (2) Even if we can employ the MapReduce model, the filtering operation based on FHE comparison becomes a significant bottleneck due to its slow performance. It involves FHE subtraction, FHE most significant bit (MSB) extraction, FHE transformation to generate an encrypted 0/1 vector filter, and vector multiplication between the filter and the data entries [2]. This process is significantly time-consuming, approximately taking 99.96% of the overall time, and it is impossible to filter out unnecessary data based on the encrypted result filter, resulting in the need to process the entire dataset.

To address these challenges, we propose HEDAS, the first distributed FHE-based OLAP system in the MapReduce style that enables secure and efficient OLAP operations. Our first insight is that by employing a pre-group operation which splits the query based on grouping attributes before assigning map tasks, HEDAS can effectively overcome the limitation of FHE filtering that doesn't support shuffling on encrypted intermediate results. By introducing two MapReduce-like stages, *Secure Map (SMap)* and *Secure Reduce (SReduce)*, HEDAS securely and efficiently executes filtering and aggregation operations in OLAP. To address the performance bottleneck associated with filtering, our second insight is to employ a design that separates data retrieval and aggregation computations. This design was inspired by the recognition that not all scenarios require the highest level of security assurance and that trade-offs between security and cost are necessary. We conducted two case studies utilizing widely accepted methods, CryptDB's index and a kind of HashFilter index, as alternatives to the FHE comparison indexing method to explore the optimization of the filter bottleneck.

We built HEDAS on top of HE^{3}DB [2], a state-of-the-art single-machine FHE-based encrypted database framework that enables secure OLAP querying. To evaluate the performance of HEDAS, we conducted experiments using the TPC-H benchmark [10]. The evaluation results demonstrate the following:

- HEDAS is efficient. It significantly reduces end-to-end latency by 44.1% compared to HE^{3}DB with four computing nodes. Additionally, the two case-study systems, HEDAS-CryptDB and HEDAS-HashFilter, achieve further 96.7% and 97.1% reductions in latency respectively, effectively mitigating the performance bottleneck of filtering.
- HEDAS is scalable. The improvements become more pronounced as the number of computing nodes increases, benefiting from the distributed computing capabilities of the MapReduce model.

In summary, our work makes the following contributions: (1) We introduce the pre-group operation and SMap/SReduce stages with FHE and OLAP co-design, effectively enabling distributed computation in FHE-based OLAP systems. (2) We present an index-computation separation architecture and conduct two case studies to address the filtering bottleneck. (3) Through evaluation experiments, we demonstrate the significant latency improvements of HEDAS compared to the state-of-the-art single-machine FHE-based OLAP system.

2 Background and Related Work

2.1 Homomorphic Encryption (HE)

Homomorphic Encryption (HE) [35] is a cryptographic technique that enables computations on encrypted data without decryption. In general, HE involves an encryption function $E(m)$ that encrypts a message m into a ciphertext c and a decryption function $D(c)$ that decrypts c back into the original m. The homomorphic property enables the computation of a function f on encrypted data c_1 and c_2, resulting in an encrypted result $f(c_1, c_2)$ which is equal to $E(g(m_1, m_2))$, where g is the corresponding function that operates on plaintext.

HE can be categorized into three types: partially homomorphic encryption (PHE), somewhat homomorphic encryption (SWHE), and fully homomorphic encryption (FHE). PHE supports only one type of operation (addition or multiplication), while SWHE supports both types of operation but with limited depth. FHE, on the other hand, supports an arbitrary number of addition and multiplication computations on encrypted data, making it the most powerful and versatile type of HE.

Since the introduction of FHE [14], numerous FHE schemes with distinct characteristics have been proposed [4,5,6,13,15]. There are also many open-source libraries that implement some of the FHE types mentioned above, such as Microsoft SEAL [37], IBM HElib [18], and TFHEpp [24]. However, FHE introduces significant computational complexity ($\sim 10^4$-$10^5\times$) and memory overhead ($\sim 10^3$-$10^4\times$), limiting its practical application in real-world scenarios.

Table 1: Comparison of HEDAS and related secure OLAP systems. **Pattern-Security** indicates whether the system can protect data access patterns, **HW-Independent** indicates whether the system is independent of specific hardware security features (e.g., TEEs), and ✗* denotes partial support for pattern security.

System	FHE-Support	Distributed	Pattern-Security	HW-Independent
CryptDB [32]	✗	✗	✗	✓
Opaque [44]	✗	✓	✗*	✗
Arx [31]	✗	✓	✗*	✓
Seabed [28]	✗	✓	✗*	✓
SAGMA [17]	✗	✓	✗*	✓
HEDA [34]	✓	✗	✓	✓
HE^3DB [2]	✓	✗	✓	✓
HEDAS	✓	✓	✓	✓

2.2 Secure OLAP Systems

We categorize the existing secure OLAP systems into three categories: FHE-based systems, systems based on specific cryptographic primitives, and TEE-based systems. In Table 1, we compare HEDAS with these systems.

FHE-based Secure OLAP Systems. FHE-based secure OLAP systems [2,34] leverage the power of FHE to enable the execution of complex analytical queries on encrypted data while preserving data confidentiality. These systems provide robust protection against data leakage and even safeguard data access patterns, owing to the strong security guarantees provided by FHE. However, the performance of these systems is limited by the computational complexity and memory overhead of FHE, and they only focus on single-machine scenarios, which restricts their scalability and efficiency when handling large datasets.

Secure OLAP Systems Based on Specific Cryptographic Primitives. In addition to FHE, other cryptographic techniques can be used for secure OLAP computations, such as Deterministic Encryption (DET), Order-Preserving Encryption (OPE)/Order-Revealing Encryption (ORE), Searchable Encryption (SE), and Partial Homomorphic Encryption (PHE). By leveraging these cryptographic primitives, it is also possible to build secure OLAP systems capable of performing queries on encrypted data (e.g., CryptDB [32]). However, these encryption schemes have security limitations that may result in data access pattern leakage [19,20], and they have limitations in scenarios involving complex computation schemes on encrypted data. Although the successors, such as Arx [31], Seabed [28] and SAGMA [17], have addressed some of these issues, they still cannot provide the same level of strong security guarantees as FHE.

TEE-based Secure OLAP Systems. Another approach to secure OLAP is leveraging Trusted Execution Environments (TEEs) provided by specific hardware. TEEs, such as Intel SGX [9] and ARM TrustZone [30], create secure enclaves within the hardware to protect sensitive data and computations. Secure OLAP systems that utilize TEEs, such as Opaque [44], provide high expressiveness with moderate performance overhead. However, TEEs are susceptible to side-channel attacks [25,22,12,39], which can be exploited by malicious insiders. Additionally, TEEs face challenges related to portability, as different hardware platforms have varying TEE implementations. Therefore, the limitations of TEEs restrict their adoption in high-security OLAP systems.

2.3 FHE Acceleration Techniques

In recent years, significant efforts have been made to accelerate FHE computations using various techniques, resulting in notable performance improvements, including methods based on GPUs [40,43], FPGAs [27], and ASICs [33,36]. We argue that these acceleration methods are complementary to our approach and can be combined to enhance the performance of FHE. On one hand, while accelerators like GPUs provide increased processing power, their capabilities are still limited. Therefore, distributed computing methods are necessary to meet the demands of large-scale data processing in OLAP. On the other hand, for each node participating in distributed OLAP computations, we can choose the appropriate acceleration methods to improve performance. In this work, we focused on optimizing FHE-based OLAP through distributed computing and didn't take these acceleration techniques into consideration.

3 Overview

3.1 System Model

Inspired by the MapReduce model, the HEDAS system also consists of three primary components: the client, the coordinator, and the worker, as depicted in Figure 1 and Figure 2. The coordinator and worker nodes are deployed in the cloud, with a single coordinator overseeing task scheduling and multiple workers dedicated to parallel task processing. Each cloud machine stores a portion of the encrypted data and can deploy multiple workers to leverage multi-core parallelism. Specifically, each type of component is described as follows:

Client. The client is the owner of the original plaintext dataset and holds the private key. Its responsibilities include encrypting the dataset, generating the encrypted attributes table, and uploading them to the coordinator. It also encrypts the query parameters, sends the query request to the coordinator, and decrypts the encrypted query results returned by the coordinator.

Coordinator. The coordinator is responsible for three main tasks: (1) Horizontally sharding the encrypted dataset and distributing it across multiple cloud machines. (2) Conducting pre-group operations based on the query and the attributes table. (3) Assigning tasks to worker nodes.

Worker. The worker nodes are responsible for processing the tasks assigned by the coordinator, which involves filtering and aggregating the encrypted data based on the query parameters.

3.2 Threat Model

Our security goal is to protect the security of the outsourced dataset in an honest-but-curious public cloud environment, and our threat model is similar to previous work [2,34,17,32], achieving the same level of security as HE^3DB [2]. We assume that the cloud service platform honestly follows our workflow to execute computations, but it may attempt to infer as much security information as possible from the dataset within a probabilistic polynomial time. Additionally, we assume that the client does not disclose any information or private keys. In our model, we categorize the information as public or confidential, as follows:

- Public information: the encrypted dataset, the encrypted attributes table, evaluation keys, the size of the dataset, the encrypted query parameters, and the query pattern (including aggregation function types, the number of query predicates, and the logical connections between predicates).
- Confidential information: the private key, the original plaintext dataset, and the plaintext query parameters.

3.3 The Pre-Group Operation

Traditionally, in the MapReduce model, the `GROUP BY` operation is performed after the map operations and before the reduce operations, which is known as data shuffle. However, when using FHE, the intermediate results generated by Map workers are encrypted randomly, making it impossible to determine which group the data belongs to. As a result, the traditional data shuffle approach is not feasible in FHE-based systems.

To address this issue, we propose the pre-group operation, conducted by the coordinator, which splits the query with the `GROUP BY` operation into numerous subqueries based on the attributes table. The attributes table is an encrypted set-like structure generated by the client, which contains all the distinct values of the grouping attribute. Each subquery replaces the `GROUP BY` operation with a homomorphic equal comparison corresponding to a specific group. By introducing the pre-group operation, the `GROUP BY` operation is performed before assigning the map tasks, resulting in a change in the shuffle model: the intermediate results produced by a Map worker now correspond to a single group, referred to as *Secure Shuffle (SShuffle)*.

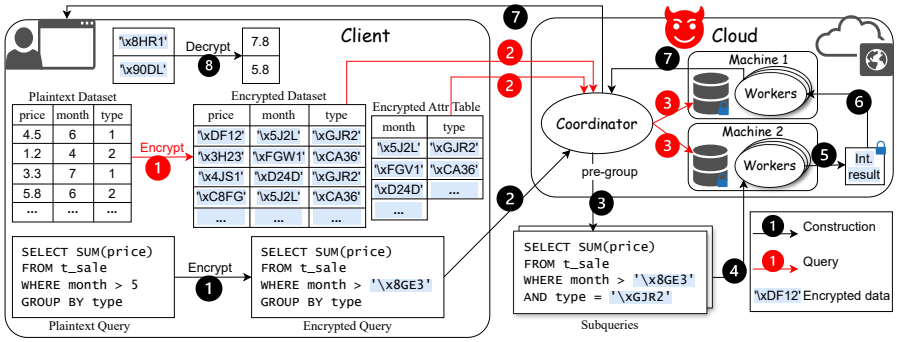


Fig. 2: The system workflow of HEDAS.

We argue that the pre-group operation does not leak the data access pattern, because the `GROUP BY` attribute values are FHE-encrypted in the attributes table, and each query with a `GROUP BY` operation is split into a predetermined number of subqueries using the same attributes table. Consequently, attacks based on data access pattern analysis [19,20] are ineffective against this scheme.

3.4 SMap and SReduce

The FHE-based OLAP process can be logically divided into two main steps: filtering and aggregation. During the filtering step, data is retrieved based on the `WHERE` conditions. However, unlike traditional OLAP indexing, FHE-based filtering cannot reduce the amount of data that needs to be processed. Instead, as shown in Figure 3, it generates a filter represented as an encrypted 0/1 vector to indicate whether the data satisfies the `WHERE` conditions. Then the data vector needs to be multiplied with the filter to aggregate the portion that satisfies the `WHERE` conditions. Therefore, shuffling the intermediate result immediately after the filtering step would require transmitting unnecessary data, resulting in increased communication overhead.

To address this communication overhead, we reorganize the processing into two stages *Secure Map (SMap)* and *Secure Reduce (SReduce)*. We first divide the aggregation into two parts: pre-aggregation and post-aggregation. Pre-aggregation involves locally aggregating the partially filtered results, reducing the amount of data transferred without increasing the computational complexity. By combining filtering and pre-aggregation, we create the SMap stage. Post-aggregation occurs after data shuffling and is responsible for aggregating the intermediate pre-aggregated results, forming the SReduce stage.

4 System Workflow

As depicted in Figure 2, the HEDAS system consists of two main phases: system construction and query processing. The system construction phase aims to

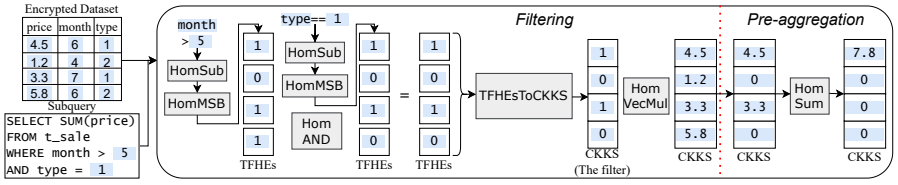


Fig. 3: The processing details of an SMap task.

Algorithm 1: System Construction

Input: Original plaintext dataset D_{pl}
// Step ❶: Encrypt dataset and generate attributes table
$pk \leftarrow$ `GetEncryptionKey()`; $ek \leftarrow$ `GenerateEvalKey(`pk`)`;
$D_{en} \leftarrow$ `Encrypt(`D_{pl},pk`)`; $attr\_table \leftarrow$ `GenerateAttributeTable(`D_{pl},pk`)`;
// Step ❷: Send encrypted data and attributes table to coordinator
`SendInitData(`$D_{en}, attr\_table, ek$`)`;
// Step ❸: Shard encrypted data and distribute to cloud machines
$machine\_num \leftarrow$ `GetMachineNum()`;
$shards\_array[] \leftarrow$ `DataSharding(`D_{en},$machine\_num$`)`;
For each cloud machine m_i**; do**
  `SendShard(`$shards\_array[m_i]$`)`;

establish a secure OLAP system on the public cloud and prepare the sharded encrypted data. Meanwhile, the query processing phase is responsible for executing secure queries and returning the encrypted query results.

4.1 System Construction

The system construction phase comprises 3 steps, as outlined by the red marks in Figure 2 and Algorithm 1:

In the system construction phase, the client first encrypts the data and generates the encrypted attributes table (step ❶), then uploads them to the coordinator on the cloud (step ❷). The attributes table is an encrypted structure consisting of all the distinct attribute values, which is utilized for the pre-group operation described in §3.3. After receiving the encrypted data and the attributes table, the coordinator horizontally partitions the encrypted data into data shards and distributes them across multiple machines (step ❸).

4.2 Query Processing

The query processing phase consists of 8 steps, as outlined by the black marks in Figure 2 and Algorithm 2:

During the query processing phase, the client first encrypts the query parameters (step ❶) and sends the encrypted query to the coordinator (step ❷). The coordinator then performs pre-group operations (step ❸) to generate subqueries based on the `GROUP BY` attribute and the corresponding attribute table. For more details on pre-group, please refer to §3.3.

Algorithm 2: Query Processing

Input: Plain text query scheme QS with plaintext filtering predicate parameters P_{pl} and `GROUP BY` attribute G

```
// Step ❶: Encrypt query parameters
pk ← GetEncryptionKey(); P_en ← Encrypt(P_pl,pk);
// Step ❷: Send query with encrypted parameters to coordinator
SendQuery(QS, P_en, G);
// Step ❸: pre-group operation
attr_table ← GetAttributeTable();
sub_queries[] ← PreGroup(QS, P_en, G, attr_table);
// Step ❹: Assign SMap tasks to worker nodes
While SMap not finished; do
    worker ← WaitWorker(); SMap_task ← GetSMapTask(sub_queries);
    AssignTask(worker, SMap_task);
// Step ❺: Workers process SMap tasks
SMap_task ← RequestSMapTask();
int_result ← DoSMapTask(SMap_task); CacheIntResult(int_result);
// Step ❻: Assign SReduce tasks
While SReduce not finished; do
    worker ← WaitWorker(); SReduce_task ← GetSReduceTask();
    AssignTask(worker, SReduce_task);
// Step ❼: Workers process SReduce tasks
SReduce_task ← RequestSReduceTask();
int_results ← GetIntResults(SReduce_task.key);
partial_result ← DoSReduceTask(SReduce_task, int_results);
SendResultToCoordinator(partial_result);
// Step ❽: Coordinator collects and returns the final result
result ← CollectAllResults(); ReturnResult(result);
```

After the pre-group separation, the coordinator assigns tasks to worker nodes, which are divided into two stages: SMap and SReduce. The details of the SMap and SReduce stages are described in §3.4.

In the SMap stage, the coordinator assigns the data shard addresses and sub-query parameters (including the specially encrypted `GROUP BY` attribute) to the map workers (step ❹). Each worker performs the filtering and pre-aggregation operations on the assigned data shard and caches the intermediate results (step ❺). To minimize communication overhead, the coordinator attempts to assign the data shard on the same machine. Once the workers complete their tasks, they inform the coordinator with the addresses of the cached intermediate results.

In the SReduce stage, the coordinator assigns the post-aggregation tasks to the reduce workers (step ❻), which is similar to the reduce operation in the MapReduce model. In SReduce, the key argument corresponds to the encrypted `GROUP BY` attribute, and the value argument represents the intermediate pre-aggregated results. The reduce workers fetch the intermediate results from the map workers as part of the SShuffle process and perform the post-aggregation operation. After all the workers complete their SReduce tasks, the coordinator collects the final results and sends them back to the client (step ❼).

After all, the client decrypts and obtains the final query results (step ❽).

5 Case Studies

As mentioned in §3.4, the OLAP process can be logically divided into two stages: indexing/filtering data and aggregation computation. Traditionally, as depicted in Figure 3, FHE-based filtering involves utilizing TFHE [7] for homomorphic subtraction on all data entries, performing homomorphic most significant bit (MSB) extraction [2], conducting homomorphic bitwise AND operations, transforming the results into CKKS [5] format to generate an encrypted 0/1 vector filter (i.e., 'TFHEsToCKKS' in Figure 3), and performing homomorphic vector multiplication between the filter and the data. This process is highly time-consuming and acts as a significant performance bottleneck, taking up approximately 99.96% of the overall time, although it provides exceptional security guarantees.

We argue that not all scenarios require such a high computational cost to maintain security. In cases where accessing the auxiliary database for attacks is challenging, it is feasible to adopt widely accepted indexing schemes, such as CryptDB-like schemes or HashFilter schemes. While these schemes may result in some access pattern leakage, they can significantly reduce the filtering cost. Therefore, we conducted two case studies to replace the FHE-based filtering scheme in HEDAS with CryptDB's indexing method and a kind of HashFilter-based indexing method to alleviate the performance bottleneck.

5.1 Case Study 1: CryptDB-based Indexing

CryptDB [32] is a secure OLAP system that relies on specific cryptographic primitives. Its indexing mechanism utilizes DET, OPE, and SE within an onion-like multi-layered encryption model to automatically adapt to various query requirements. In our case study, we applied CryptDB's onion-like indexing method to HEDAS, replacing the FHE-based filtering approach. This modified system is referred to as HEDAS-CryptDB. The workflow of HEDAS-CryptDB is mostly identical to that of HEDAS, with the pre-group operation removed and regular data shuffle employed, because the use of DET and OPE allows for conventional data shuffling.

5.2 Case Study 2: HashFilter-based Indexing

HashFilter-based indexing is a straightforward and efficient method that uses a hash function to map data to a fixed-size space for rapid data retrieval, and further enhances the indexing process by employing filters (e.g., Bloom filter [3]) to quickly eliminate data that does not meet specific conditions. Drawing on previous research [16,29], we propose a tree-like index structure based on the one-way hash function and the Bloom filter to accelerate the retrieval of encrypted data. To address the issue of false positives caused by the Bloom filter, we introduce a timely verification mechanism based on the characteristics of the tree structure to promptly eliminate such occurrences. In this case study, we integrate this indexing method into HEDAS, replacing the FHE-based filtering

Table 2: Latency breakdown analysis for 4 systems with 256 rows, where N/A indicates not applicable.

System	Filtering(s)	Aggregation(ms)		Shuffle(ms)	Total Latency(s)
		Pre-aggregation(ms)	Post-aggregation(ms)		
HE^{3}DB	286.59(±0.60)	87.35(±0.14)		N/A	286.67(±0.60)
Hedas	154.39(±25.36)	67.90(±5.25)	2.12(±0.02)	65.09(±2.25)	168.29(±27.89)
Hedas-CryptDB	0.81(±0.04)	66.46(±3.22)	0.90(±0.40)	60.92(±4.06)	6.80(±1.24)
Hedas-HashFilter	0.53(±0.06)	69.94(±6.16)	1.15(±0.47)	64.67(±4.22)	6.72(±0.30)

approach, referred to as Hedas-HashFilter. The workflow of Hedas-HashFilter closely resembles that of Hedas-CryptDB.

6 Evaluation

6.1 Experiment Setup

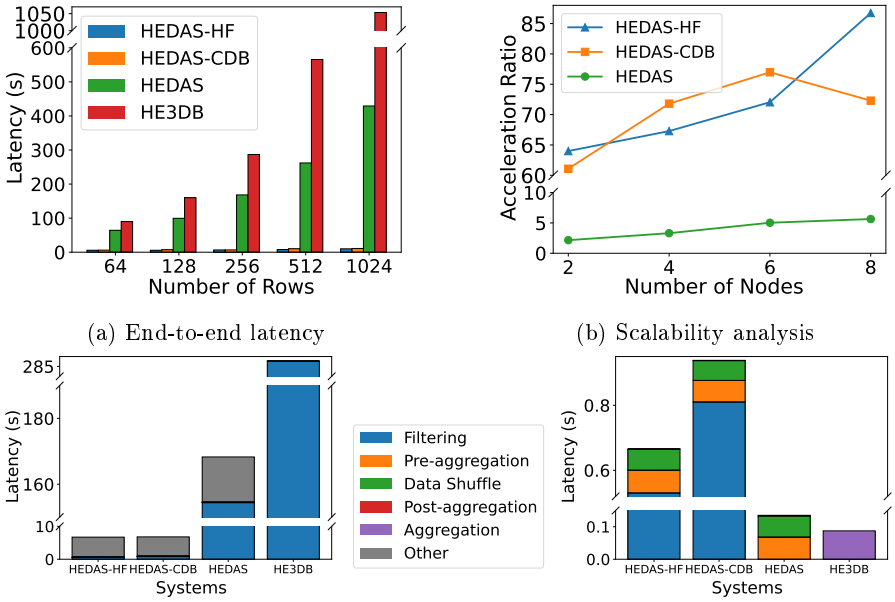
We implemented Hedas using the HE^{3}DB [2] framework and utilized the Microsoft SEAL [37], OpenPEGASUS [23], and TFHEpp [24] libraries for specific functions. We then conducted two case studies on the Hedas system. Our compilation environment was GCC 11.4.0 with the `-O3` optimization level and `C++17` standard. The runtime environment consisted of 10 machines with identical configurations, each equipped with a 2.60GHz Intel E5-2690 V3 CPU, 64GB memory, 40Gbps NIC, and 24 cores. The average ping latency between the nodes was ∼0.102ms. We used the same benchmark as HE^{3}DB, which is a subset of the TPC-H [10] queries, and adopted identical encryption parameters as HE^{3}DB, ensuring that the encryption overhead mirrored that of HE^{3}DB. For latency, we utilize the same metric as HE^{3}DB, which logs the timestamp of critical points in the query process and calculates the time span for each part.

Throughout our experiments, we aim to explore the following three research questions:

- To what extent can Hedas's performance be improved compared to the state-of-the-art single-machine system?
- How much additional performance improvement can be achieved by changing the filtering method?
- How does Hedas's performance vary as the number of computing nodes increases?

6.2 End-to-End Performance

We evaluated to compare the performance of Hedas, Hedas-CryptDB, and Hedas-HashFilter on 4 computing nodes, using HE^{3}DB running on a single machine as the baseline, and the TPC-H query 6 was used as the test case.



(a) End-to-end latency

(b) Scalability analysis

(c) Breakdown latency

(d) Other breakdown parts

Fig. 4: Evaluation results of HEDAS, HEDAS-CryptDB (CDB), HEDAS-HashFilter (HF), and HE^{3}DB.

The results, as illustrated in Figure 4a, demonstrate that HEDAS achieves significantly lower latency compared to HE^{3}DB, ranging from 40.8% to 71.6%. HEDAS-CryptDB and HEDAS-HashFilter further reduce the latency, achieving 2.56% to 9.47% and 2.33% to 8.97% of HEDAS's latency, respectively. These findings strongly indicate that HEDAS outperforms HE^{3}DB in terms of end-to-end latency, and additional performance improvements can be achieved by changing the indexing method, although they trade off security guarantees.

6.3 Breakdown Analysis

To further conduct a comprehensive analysis of the performance improvements, we performed a latency breakdown analysis on a dataset of 256 rows using 4 nodes across the 4 systems, as illustrated in Figure 4c and summarized in Table 2. For HEDAS, HEDAS-CryptDB, and HEDAS-HashFilter, we divided the query process into four parts: filtering, pre-aggregation, shuffle, and post-aggregation. Additionally, there is also an 'other' part, including time for waiting, data loading, and other overheads. In the case of HE^{3}DB, the query process was divided into two parts: filtering and aggregation. Notably, Table 2 reveals that the Filtering part consumes significantly more time compared to other parts across all sys-

tems. Consequently, for better comparability, the breakdown of the remaining parts is also presented in Figure 4d.

The results demonstrate that when comparing HEDAS and HE^{3}DB, although HEDAS introduces an additional shuffle communication part, the parallel execution effectively distributes the computational overhead, resulting in a significant reduction in the latency of both the filtering and aggregation parts, reducing approximately 46.1% and 19.8% of HE^{3}DB's latency, respectively. Regarding the comparison between the two case study systems (i.e., HEDAS-CryptDB and HEDAS-HashFilter) and HEDAS, it is observed that the latency of the filtering part is significantly decreased in both cases, accounting for approximately 99.7% and 99.8% reduction of HEDAS's latency, respectively. Consequently, changing the filtering method proves to be an effective approach to addressing the performance bottleneck associated with FHE-based filtering.

6.4 Scalability Analysis

To evaluate the scalability of HEDAS, we conducted performance experiments on HEDAS and the two case study systems using varying numbers of computing nodes, as shown in Figure 4b.

The results demonstrate that the performance of HEDAS exhibits a clear increasing trend as the number of computing nodes increases, indicating good scalability. However, it is observed that as the number of computing nodes continues to increase, the growth rate of HEDAS's performance slightly diminishes. We attribute this phenomenon to the increasing network overhead. Regarding the two case study systems, both HEDAS-CryptDB and HEDAS-HashFilter demonstrate good scalability with similar acceleration ratios. However, their acceleration ratios show more fluctuation compared to HEDAS, potentially due to their faster but erratic speed.

7 Conclusion

In this work, we propose HEDAS, the first distributed FHE-based OLAP system in the MapReduce style, enabling secure and efficient OLAP operations. By introducing the pre-group operation and the SMap and SReduce stages, HEDAS effectively addresses the limitations of FHE in distributed OLAP scenarios. To address the performance bottleneck of FHE-based filtering, we present an index-computation separation architecture and conduct two case studies replacing the FHE-based filtering method in HEDAS. Our experimental results demonstrate that HEDAS outperforms the state-of-the-art single-machine FHE-based OLAP system by leveraging the parallel execution capabilities of the MapReduce model, and the case study systems exhibit even better performance improvements. Additionally, our scalability experiments confirm that HEDAS exhibits good scalability, further validating its potential for handling larger workloads. In conclusion, HEDAS successfully addresses the challenges of secure and efficient OLAP operations in the public cloud and provides a promising solution.

8 Acknowledgments

The work is supported in part by National Key R&D Program of China (2022ZD0160201), HK RIF (R7030-22), HK ITF (GHP/169/20SZ), the Huawei Flagship Research Grant in 2023, HK RGC GRF (Ref.: 17208223 and 17204424), National Key R&D Program of China (2023YFB4503902), the HKU-CAS Joint Laboratory for Intelligent System Software, and the Shanghai Artificial Intelligence Laboratory.

References

1. Akinde, M.O., Böhlen, M.H., Johnson, T., Lakshmanan, L.V., Srivastava, D.: Efficient olap query processing in distributed data warehouses. Information Systems **28**(1-2), 111–135 (2003)
2. Bian, S., Zhang, Z., Pan, H., Mao, R., Zhao, Z., Jin, Y., Guan, Z.: He3db: An efficient and elastic encrypted database via arithmetic-and-logic fully homomorphic encryption. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. pp. 2930–2944 (2023)
3. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7), 422–426 (1970)
4. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT) **6**(3), 1–36 (2014)
5. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic encryption for arithmetic of approximate numbers. In: Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. pp. 409–437. Springer (2017)
6. Chillotti, I., Gama, N., Georgieva, M., Izabachene, M.: Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In: Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22. pp. 3–33. Springer (2016)
7. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (2020)
8. Codd, E.F.: Providing olap (on-line analytical processing) to user-analysts: An it mandate. http://www. arborsoft. com/papers/coddTOC. html (1993)
9. Costan, V., Devadas, S.: Intel sgx explained. Cryptology ePrint Archive (2016)
10. Council, T.P.P.: TPC BENCHMARK™ h standard specification. Technical report, Transaction Processing Performance Council, San Francisco, CA (2022)
11. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM **51**(1), 107–113 (2008)
12. Disselkoen, C., Kohlbrenner, D., Porter, L., Tullsen, D.: {Prime+ Abort}: A {Timer-Free}{High-Precision} l3 cache attack using intel {TSX}. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 51–67 (2017)
13. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)

14. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the forty-first annual ACM symposium on Theory of computing. pp. 169–178 (2009)
15. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. pp. 75–92. Springer (2013)
16. Goh, E.J.: Secure indexes. Cryptology ePrint Archive (2003)
17. Hackenjos, T., Hahn, F., Kerschbaum, F.: Sagma: secure aggregation grouped by multiple attributes. In: Proceedings of the 2020 ACM SIGMOD international conference on management of data. pp. 587–601 (2020)
18. IBM: HElib. https://github.com/homenc/HElib (2021), accessed: 2024-5-28
19. Islam, M.S., Kuzu, M., Kantarcioglu, M.: Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In: Ndss. vol. 20, p. 12. Citeseer (2012)
20. Kellaris, G., Kollios, G., Nissim, K., O'neill, A.: Generic attacks on secure outsourced databases. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. pp. 1329–1340 (2016)
21. Latreche, O., Boukraa, D.: Self-service, on-demand creation of olap cubes over big data: a metadata-driven approach. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 2907–2914. IEEE (2020)
22. Lee, S., Shih, M.W., Gera, P., Kim, T., Kim, H., Peinado, M.: Inferring fine-grained control flow inside {SGX} enclaves with branch shadowing. In: 26th USENIX Security Symposium (USENIX Security 17). pp. 557–574 (2017)
23. Lu, W.j., Huang, Z., Hong, C., Ma, Y., Qu, H.: Pegasus: bridging polynomial and non-polynomial evaluations in homomorphic encryption. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1057–1073. IEEE (2021)
24. Matsuoka, K.: TFHEpp: pure C++ implementation of TFHE cryptosystem. https://github.com/virtualsecureplatform/TFHEpp (2020)
25. Moghimi, A., Irazoqui, G., Eisenbarth, T.: Cachezoom: How sgx amplifies the power of cache attacks. In: Cryptographic Hardware and Embedded Systems–CHES 2017: 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. pp. 69–90. Springer (2017)
26. Mulazzani, M., Schrittwieser, S., Leithner, M., Huber, M., Weippl, E.: Dark clouds on the horizon: Using cloud storage as attack vector and online slack space. In: 20th USENIX Security Symposium (USENIX Security 11) (2011)
27. Nam, K., Oh, H., Moon, H., Paek, Y.: Accelerating n-bit operations over tfhe on commodity cpu-fpga. In: Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design. pp. 1–9 (2022)
28. Papadimitriou, A., Bhagwan, R., Chandran, N., Ramjee, R., Haeberlen, A., Singh, H., Modi, A., Badrinarayanan, S.: Big data analytics over encrypted datasets with seabed. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16). pp. 587–602 (2016)
29. Pappas, V., Krell, F., Vo, B., Kolesnikov, V., Malkin, T., Choi, S.G., George, W., Keromytis, A., Bellovin, S.: Blind seer: A scalable private dbms. In: 2014 IEEE Symposium on Security and Privacy. pp. 359–374. IEEE (2014)
30. Pinto, S., Santos, N.: Demystifying arm trustzone: A comprehensive survey. ACM computing surveys (CSUR) **51**(6), 1–36 (2019)
31. Poddar, R., Boelter, T., Popa, R.A.: Arx: An encrypted database using semantically secure encryption. Cryptology ePrint Archive (2016)

32. Popa, R.A., Redfield, C.M., Zeldovich, N., Balakrishnan, H.: Cryptdb: Protecting confidentiality with encrypted query processing. In: Proceedings of the twenty-third ACM symposium on operating systems principles. pp. 85–100 (2011)
33. Reagen, B., Choi, W.S., Ko, Y., Lee, V.T., Lee, H.H.S., Wei, G.Y., Brooks, D.: Cheetah: Optimizing and accelerating homomorphic encryption for private inference. In: 2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA). pp. 26–39. IEEE (2021)
34. Ren, X., Su, L., Gu, Z., Wang, S., Li, F., Xie, Y., Bian, S., Li, C., Zhang, F.: Heda: multi-attribute unbounded aggregation over homomorphically encrypted database. Proceedings of the VLDB Endowment **16**(4), 601–614 (2022)
35. Rivest, R.L., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
36. Samardzic, N., Feldmann, A., Krastev, A., Devadas, S., Dreslinski, R., Peikert, C., Sanchez, D.: F1: A fast and programmable accelerator for fully homomorphic encryption. In: MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture. pp. 238–252 (2021)
37. Microsoft SEAL (release 3.7). https://github.com/Microsoft/SEAL (Sep 2021), microsoft Research, Redmond, WA.
38. of Standards, N.I., Technology: Advanced encryption standard. NIST FIPS PUB 197 (2001)
39. Wang, W., Chen, G., Pan, X., Zhang, Y., Wang, X., Bindschaedler, V., Tang, H., Gunter, C.A.: Leaky cauldron on the dark land: Understanding memory side-channel hazards in sgx. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 2421–2434 (2017)
40. Wang, Z., Li, P., Hou, R., Li, Z., Cao, J., Wang, X., Meng, D.: He-booster: an efficient polynomial arithmetic acceleration on gpus for fully homomorphic encryption. IEEE Transactions on Parallel and Distributed Systems **34**(4), 1067–1081 (2023)
41. Xiao, L., Xu, D., Xie, C., Mandayam, N.B., Poor, H.V.: Cloud storage defense against advanced persistent threats: A prospect theoretic study. IEEE Journal on Selected Areas in Communications **35**(3), 534–544 (2017)
42. Xue, K., Chen, W., Li, W., Hong, J., Hong, P.: Combining data owner-side and cloud-side access control for encrypted cloud storage. IEEE Transactions on Information Forensics and Security **13**(8), 2062–2074 (2018)
43. Yang, H., Shen, S., Dai, W., Zhou, L., Liu, Z., Zhao, Y.: Phantom: A cuda-accelerated word-wise homomorphic encryption library. IEEE Transactions on Dependable and Secure Computing (2024)
44. Zheng, W., Dave, A., Beekman, J.G., Popa, R.A., Gonzalez, J.E., Stoica, I.: Opaque: An oblivious and encrypted distributed analytics platform. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17). pp. 283–298 (2017)

Card-based Cryptographic Protocols for Three-input Functions with a Standard Deck of Cards Using Private Operations

Naoki Kobayashi[1] and Yoshifumi Manabe[1 [0000−0002−6312−257X]]

School of Informatics, Kogakuin University,
Shinjuku, Tokyo 163-8677 Japan. manabe@cc.kogakuin.ac.jp

Abstract. This paper shows card-based cryptographic protocols to calculate several Boolean functions with a standard deck of cards using private operations. They are multi-party secure computations executed by multiple semi-honest players without computers. The protocols use private operations that are executed by a player at a place where the other players cannot see. Most card-based cryptographic protocols use a special deck of cards that consists of many cards with two kinds of marks. Though these protocols are simple and efficient, the users need to prepare such special cards. Few protocols were shown that use a standard deck of playing cards, though the protocols with a standard deck of cards can be easily executed in our daily lives. It was shown that logical AND, logical XOR, and copy protocols can be executed with the minimum number of cards. However, the protocols for complicated functions are not known. This paper shows that by using private operations, all of the following Boolean functions can be calculated without additional cards other than the input cards: (1) any three input Boolean functions, (2) half adder and full adder, and (3) any n-input symmetric Boolean functions. The results show the effectiveness of private operations in card-based cryptographic protocols.

Keywords: card-based cryptographic protocols · multi-party secure computation · Boolean functions · half adder · symmetric functions · private operations · standard deck of cards.

1 Introduction

1.1 Overview of Card-based Cryptographic Protocols

Card-based cryptographic protocols [26, 28] were proposed in which physical cards are used instead of computers to securely compute values. They can be used when computers cannot be used or users cannot trust the software on the computer. Also, the protocols are easy to understand, thus the protocols can be used to teach the basics of cryptography [4, 21] to accelerate the social implementation of advanced cryptography [6]. den Boer [3] first showed a five-card protocol to securely compute the logical AND of two inputs. Since then,

many protocols have been proposed to realize primitives to compute any Boolean functions [8, 11, 29, 34, 37, 38, 47, 48] and computate a specific class of Boolean functions [1, 2, 5, 7, 13–15, 18, 22, 25, 35, 36, 40, 41, 44–46, 50, 51].

This paper considers computations of (1) any three input Boolean functions, (2) half adder and full adder, and (3) any n-input symmetric Boolean functions. No additional cards are necessary to calculate these functions with a standard deck of cards when we use private operations.

Note that in this paper, all players are assumed to be semi-honest. Few works are done for the case when some players are malicious or make mistakes [10, 16, 24, 27, 30, 49].

1.2 Standard Deck of Cards

Most of the above works are based on a two-color card model. In the two-color card model, there are two kinds of cards, $\boxed{\clubsuit}$ and $\boxed{\heartsuit}$. Cards of the same marks cannot be distinguished. In addition, the back of both types of cards is $\boxed{?}$. It is impossible to determine the mark in the back of a given card of $\boxed{?}$. Though the model is simple, such special cards are not available in our daily lives.

To solve the problem, card-based cryptographic protocols using a standard deck of playing cards were shown [9, 12, 13, 19, 20, 23, 30, 31, 33, 47]. Playing cards are available at many houses and are easy to buy. The standard deck of playing cards is also used for zero-knowledge proof of puzzle solutions [39, 42]. This paper discusses protocols to calculate logical functions. Niemi and Renvall first showed protocols that use a standard deck of playing cards [33]. They showed logical XOR, logical AND, and copy protocols since any Boolean functions can be realized by a combination of these protocols. Their protocols are 'Las Vegas' type protocols, that is, the execution times of the protocols are not limited. The protocols are expected to terminate within a finite time and the efficiency of these protocols is evaluated by the expected execution time. However, if the sequence of the random numbers is bad, the protocols do not terminate forever. Mizuki showed fixed time logical XOR, logical AND, and copy protocols [23]. Though the number of cards used by the XOR protocol is the minimum, the ones used by the logical AND and copy protocols are not the minimum. Koch et al. showed a four-card 'Las Vegas' type AND protocol and it is impossible to obtain a four-card finite time protocol with the model without private operations [9]. Koyama et al. showed a three-input 'Las Vegas' type AND protocol with the minimum number of cards [12]. Koyama et al. showed an efficient 'Las Vegas' type copy protocol [13]. Shinagawa and Mizuki showed protocols to compute any n-variable function using a standard deck of playing cards and a deck of UNO[1] cards [47]. Miyahara et al. showed a protocol that solves Yao's millionares' problem using a standard deck of playing cards [19]. Miyahara and Mizuki showed new protocols that use a special primitive that opens the suit of a playing card [20]. This paper discusses protocols that publically or privately open the cards. Another class of protocols is considered, in which each player knows his/her private data, and

[1] https://www.letsplayuno.com

the player privately inputs the data to the protocol. Nakai et al. showed AND, XOR, and majority protocols [31].

1.3 Private Operations

Randomization or a private operation is the most important primitive in these card-based protocols. If every primitive executed in a card-based protocol is deterministic and public, the relationship between the private input values and the output values is known to the players. When the output values are disclosed, the private input values can be known to the players from the relationship. Thus, all protocols need some random or private operation.

First, public randomization primitives have been discussed then recently, private operations are considered. Many protocols use random bisection cuts [29], which randomly execute swapping two decks of cards or not swapping. If the random value used in the randomization is disclosed, the secret input value is known to the players. If some player privately brings a high-speed camera, the random value selected by the randomization might be known by analyzing the image. Though the size of a high-speed camera is very large, the size might become very small shortly. To prepare for the situation, we need to consider using private operations.

Operations that a player executes in a place where the other players cannot see are called private operations. These operations are considered to be executed under the table or in the back. Private operations are shown to be the most powerful primitives in card-based cryptographic protocols. They were first introduced to solve the millionaires' problem [32]. Using three private operations shown later, committed-input and committed-output logical AND, logical XOR, and copy protocols can be achieved with the minimum number of cards on the two-color card model [37].

For the primitives of logical AND, logical XOR, and copy operation, the minimum number of cards is achieved with a standard deck of cards using private operations [17]. So the research question is whether we can achieve the minimum number of cards for complicated calculations.

1.4 Our Results

This paper shows new card-based protocols with a standard deck of cards using private operations to calculate (1) any three input Boolean functions, (2) half adder and full adder, and (3) any n-input symmetric Boolean functions. All of these protocols need no additional cards other than the input cards. Thus these protocols are optimal in regard to the number of cards.

In Section 2 basic definitions and the private operations introduced by [37] are shown. Then, the sub-protocols shown in [17] and used in this paper are stated. Section 3 shows protocols to calculate three input Boolean functions. Section 4 shows protocols to calculate half and full adder, and n-input symmetric Boolean functions. Section 5 concludes the paper.

2 Preliminaries

2.1 Basic Notations

This section gives the notations and basic definitions of card-based protocols with a standard deck of cards. A deck of playing cards consists of 52 distinct mark cards, which are named as 1 to 52. The number of each card (for example, 1 is the ace of spade and 52 is the king of club) is common knowledge among the players. The back of all cards is the same $\boxed{?}$. It is impossible to determine the mark in the back of a given card of $\boxed{?}$.

One-bit data is represented by two cards as follows: $\boxed{\mathbf{i}}\boxed{\mathbf{j}} = 0$ and $\boxed{\mathbf{j}}\boxed{\mathbf{i}} = 1$ if $i < j$.

One pair of cards that represents one bit $x \in \{0, 1\}$, whose face is down, is called a commitment of x, and denoted as $commit(x)$. It is written as $\underset{x}{\boxed{?}\boxed{?}}$. The base of a commitment is the pair of cards used for the commitment. If card i and $j (i < j)$ are used to set $commit(x)$ (That is, set $\boxed{\mathbf{i}}\boxed{\mathbf{j}}$ if $x = 0$ and set $\boxed{\mathbf{j}}\boxed{\mathbf{i}}$ if $x = 1$), the commitment is written as $commit(x)^{\{i,j\}}$ and written as $\underset{x^{\{i,j\}}}{\boxed{?}\boxed{?}}$. When the base information is obvious or unnecessary, it is not written.

Note that when these two cards are swapped, $commit(\bar{x})^{\{i,j\}}$ can be obtained. Thus, logical negation can be computed without private operations.

A set of cards placed in a row is called a sequence of cards. A sequence of cards S whose length is n is denoted as $S = s_1, s_2, \ldots, s_n$, where s_i is i-th card of the sequence. $S = \overset{s_1}{\boxed{?}}\ \overset{s_2}{\boxed{?}}\ \overset{s_3}{\boxed{?}}\ \ldots,\ \overset{s_n}{\boxed{?}}$. A sequence whose length is even is called an even sequence. $S_1||S_2$ is a concatenation of sequence S_1 and S_2.

All protocols are executed by two players, Alice and Bob. The players are semi-honest, that is, they obey the rules of the protocols but try to obtain secret values.

The inputs of the protocols are given in a committed manner, that is, the players do not know the input values. If a player knows his secure input value x, the player just makes a commitment of x, and the protocols in this paper can be used. The output of the protocol must be given in a committed format so that the result can be used as an input to further computation. If the players need to obtain the output value, they just open the committed output. Thus committed output is desirable.

A protocol is secure when the following two conditions are satisfied: (1) If the output cards are not opened, each player obtains no information about the private input values from the view of the protocol for the player (the sequence of the cards opened to the player). (2) When the output cards are opened, each player obtains no additional information about the private input values other than the information by the output of the protocol. For example, if the output cards of an AND protocol for input x and y are opened and the value is 1, the

players can know that $x = 1$ and $y = 1$. If the output value is 0, the players must not know whether the input (x, y) is $(0, 0)$, $(0, 1)$, or $(1, 0)$.

The following protocols use random numbers. Random numbers can be generated without computers using coin-flipping or some similar methods. During the protocol executions, cards are sent and received between the players. The communication is executed by sending the cards between the players to avoid information leakage during the communication. If the players are not in the same place during the protocol execution, a trusted third party (for example, the post office) is necessary to send and receive cards between players.

2.2 Private Operations

We show three private operations introduced in [37]: private random bisection cuts, private reverse cuts, and private reveals.

Primitive 1 *(Private random bisection cut)*

A private random bisection cut is the following operation on an even sequence $S_0 = s_1, s_2, \ldots, s_{2m}$. *A player selects a random bit* $b \in \{0, 1\}$ *and outputs*

$$S_1 = \begin{cases} S_0 & \text{if } b = 0 \\ s_{m+1}, s_{m+2}, \ldots, s_{2m}, s_1, s_2, \ldots, s_m & \text{if } b = 1 \end{cases}$$

The player executes this operation in a place where the other players cannot see. The player must not disclose the bit b.

Note that if the private random cut is executed when $m = 1$ and $S_0 = commit(x)$, given $S_0 = \underset{x}{\boxed{?}\boxed{?}}$, The player's output $S_1 = \underset{x \oplus b}{\boxed{?}\boxed{?}}$, which is $\underset{x}{\boxed{?}\boxed{?}}$ or $\underset{\overline{x}}{\boxed{?}\boxed{?}}$.

Note that a private random bisection cut is the same as the random bisection cut [29], but the operation is executed in a hidden place.

Primitive 2 *(Private reverse cut, Private reverse selection)*

A private reverse cut is the following operation on an even sequence $S_2 = s_1, s_2, \ldots, s_{2m}$ *and a bit* $b \in \{0, 1\}$. *A player outputs*

$$S_3 = \begin{cases} S_2 & \text{if } b = 0 \\ s_{m+1}, s_{m+2}, \ldots, s_{2m}, s_1, s_2, \ldots, s_m & \text{if } b = 1 \end{cases}$$

The player executes this operation in a place where the other players cannot see. The player must not disclose b.

Note that the bit b is not newly selected by the player. This is the difference between the primitive in Primitive 1, where a random bit must be newly selected by the player.

Note that in some protocols below, selecting left m cards is executed after a private reverse cut. The sequence of these two operations is called a private reverse selection. A private reverse selection is the following procedure on an even sequence $S_2 = s_1, s_2, \ldots, s_{2m}$ and a bit $b \in \{0, 1\}$. A player outputs

$$S_3 = \begin{cases} s_1, s_2, \ldots, s_m & \text{if } b = 0 \\ s_{m+1}, s_{m+2}, \ldots, s_{2m} & \text{if } b = 1 \end{cases}$$

Primitive 3 *(Private reveal) A player privately opens a given committed bit. The player must not disclose the obtained value.*

Using the obtained value, the player privately sets a sequence of cards.

Consider the case when Alice executes a private random bisection cut on $commit(x)$ and Bob executes a private reveal on the bit. Since the committed bit is randomized by the bit b selected by Alice, the opened bit is $x \oplus b$. Even if Bob privately opens the cards, Bob obtains no information about x if b is randomly selected and not disclosed by Alice. Bob must not disclose the obtained value. If Bob discloses the obtained value to Alice, Alice knows the value of the committed bit.

2.3 Opaque Commitment Pair

An opaque commitment pair is defined as a useful situation to design a secure protocol using a standard deck of cards [23]. It is a pair of commitments whose bases are unknown to a player. Let us consider the following two commitments using cards i, j, i' and j'. The left (right) commitment has value x (y), respectively, but it is unknown that (1) the left (right) commitment is made using i and j (i' and j'), respectively, or (2) the left (right) commitment is made using i' and j' (i and j), respectively. Such a pair of commitments is called an opaque commitment pair and written as $commit(x)^{\{i,j\},\{i',j'\}}||commit(y)^{\{i,j\},\{i',j'\}}$.

The protocols in this paper use a little different kind of pair, called semi-opaque commitment pair. A player thinks a pair is an opaque commitment pair but another player knows the bases of the commitments. Let us consider the case when a protocol is executed by Alice and Bob. Bob privately makes the pair of commitments with the knowledge of x and y. For example, Bob randomly selects a bit $b \in \{0, 1\}$ and

$$S = \begin{cases} commit(x)^{\{i,j\}}||commit(y)^{\{i',j'\}} & \text{if } b = 0 \\ commit(x)^{\{i',j'\}}||commit(y)^{\{i,j\}} & \text{if } b = 1 \end{cases}$$

then $S = commit(x)^{\{i,j\},\{i',j'\}}||commit(y)^{\{i,j\},\{i',j'\}}$ for Alice. Such a pair is called a semi-opaque commitment pair and written as $commit(x)^{\{i,j\},\{i',j'\}|Alice}||$ $commit(y)^{\{i,j\},\{i',j'\}|Alice}$, where the name(s) of the players who think the pair is a opaque commitment pair is written. Note that a name is not written does not mean the player knows the bases of the commitments. For example, the above example says nothing about whether Bob knows the bases or not. Note that the name of the player is written with the initial when it is not ambiguous.

2.4 Space and Time Complexities

The space complexity of card-based protocols is evaluated by the number of cards. Minimizing the number of cards is discussed in many works.

The number of rounds was proposed as a criterion to evaluate the time complexity of card-based protocols using private operations [38]. The first round begins from the initial state. The first round is (possibly parallel) local executions by each player using the cards initially given to each player. It ends at the instant when no further local execution is possible without receiving cards from another player. The local executions in each round include sending cards to some other players but do not include receiving cards. The result of every private execution is known to the player. For example, shuffling whose result is unknown to the player himself is not executed. Since the private operations are executed in a place where the other players cannot see, it is hard to force the player to execute such operations whose result is unknown to the player. The $i(>1)$-th round begins with receiving all the cards sent during the $(i-1)$-th round. Each player executes local executions using the received cards and the cards left to the player at the end of the $(i-1)$-th round. Each player executes local executions until no further local execution is possible without receiving cards from another player. The number of rounds of a protocol is the maximum number of rounds necessary to output the result among all possible inputs and random values. If the local execution needs many operations, for example, $O(n)$ operations where n is the size of the problem, we might need another criterion to consider the cost of local executions.

Let us show an example of a protocol execution, its space complexity, and time complexity.

Protocol 1 *(XOR protocol) [17]*
Input: $commit(x)^{\{1,2\}}$ *and* $commit(y)^{\{3,4\}}$.
Output: $commit(x \oplus y)^{\{1,2\}}$.

1. *Alice executes a private random bisection cut on* $commit(x)^{\{1,2\}}$ *and* $commit(y)^{\{3,4\}}$ *using the same random bit* $b \in \{0,1\}$. *The result is* $commit(x \oplus b)^{\{1,2\}}$ *and* $commit(y \oplus b)^{\{3,4\}}$. *Alice sends these cards to Bob.*
2. *Bob executes a private reveal on* $commit(y \oplus b)^{\{3,4\}}$. *Bob sees* $y \oplus b$. *Bob executes a private reverse cut on* $commit(x \oplus b)^{\{1,2\}}$ *using* $y \oplus b$. *The result is* $commit((x \oplus b) \oplus (y \oplus b))^{\{1,2\}} = commit(x \oplus y)^{\{1,2\}}$.

The number of cards is four since no cards are used other than the inputs.

Let us consider the time complexity of the protocol. The first round ends at the instant when Alice sends $commit(x \oplus b)^{\{1,2\}}$ and $commit(y \oplus b)^{\{3,4\}}$ to Bob. The second round begins with receiving the cards by Bob. The number of rounds of this protocol is two.

Since each operation is relatively simple, the dominating time to execute protocols with private operations is the time to send cards between players and set up so that the cards are not seen by the other players. Thus the number of rounds is the criterion to evaluate the time complexity of card-based protocols with private operations.

2.5 Protocols for AND, Copy, and Other Boolean Functions

This subsection shows the sub-protocols presented in [17] used in this paper's protocols. The correctness proof is shown in [17].

AND Protocol Before showing the AND protocol, a subprotocol to fix the base of commitments is shown.

Protocol 2 *(Base-fixed protocol) [17]*
Input: $commit(x)^{\{1,2\},\{3,4\}|A}||commit(y)^{\{1,2\},\{3,4\}|A}$.
(Note: y is a private value that must not be known to the players)
Output: $commit(x)^{\{1,2\}}$.

1. *Bob executes a private random bisection cut on both pairs using two distinct random bits* $br_1, br_2 \in \{0,1\}$. *The result* $S_1 = commit(x \oplus br_1)^{\{1,2\},\{3,4\}|A}||commit(y \oplus br_2)^{\{1,2\},\{3,4\}|A}$. *Bob sends* S_1 *to Alice.*
2. *Alice executes a private reveal on* S_1. *Alice sees* $x \oplus br_1$ *and* $y \oplus br_2$. *If the base of the left pair is* $\{1,2\}$, *Alice just faces down the left pair and the cards,* S_2, *are the result. Otherwise, the base of the right pair is* $\{1,2\}$. *Alice makes* $S_2 = commit(x \oplus br_1)^{\{1,2\}}$ *using the right cards. Alice sends* S_2 *to Bob.*
3. *Bob executes a private reverse cut using* br_1 *on* S_2. *The result is* $commit(x)^{\{1,2\}}$.

Using the base-fixed protocol, the AND protocol in [17] is shown below.

Protocol 3 *(AND protocol) [17]*
Input: $commit(x)^{\{1,2\}}$ *and* $commit(y)^{\{3,4\}}$.
Output: $commit(x \wedge y)^{\{1,2\}}$.

1. *Alice executes a private random bisection cut on* $commit(x)^{\{1,2\}}$ *using random bit* a_1. *Alice sends the results,* $S_1 = commit(x \oplus a_1)^{\{1,2\}}$ *and* $S_2 = commit(y)^{\{3,4\}}$ *to Bob.*
2. *Bob executes a private reveal on* S_1. *Bob sees* $x \oplus a_1$. *Bob privately sets*

$$S_{3,0} = \begin{cases} commit(0)^{\{1,2\}}||commit(y)^{\{3,4\}} & \text{if } x \oplus a_1 = 0 \\ commit(y)^{\{3,4\}}||commit(0)^{\{1,2\}} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Bob sends $S_{3,0}$ *to Alice.*
3. *Alice executes private random bisection cuts on each of pairs in* $S_{3,0}$ *using two distinct random bits* a_2 *and* a_3. *Let the result be* $S_{3,1}$.

$$S_{3,1} = \begin{cases} commit(0 \oplus a_2)^{\{1,2\}}||commit(y \oplus a_3)^{\{3,4\}} & \text{if } x \oplus a_1 = 0 \\ commit(y \oplus a_2)^{\{3,4\}}||commit(0 \oplus a_3)^{\{1,2\}} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Alice sends $S_{3,1}$ *to Bob.*
4. *Bob randomly selects bit* $b_1 \in \{0,1\}$. *Bob reveals* $S_{3,1}$ *and exchanges the bases of the two commitments if* $b_1 = 1$. *Let the result be* $S_{3,2}$.

$$S_{3,2} = \begin{cases} commit(0 \oplus a_2)^{\{1,2\},\{3,4\}|A}||commit(y \oplus a_3)^{\{1,2\},\{3,4\}|A} & \text{if } x \oplus a_1 = 0 \\ commit(y \oplus a_2)^{\{1,2\},\{3,4\}|A}||commit(0 \oplus a_3)^{\{1,2\},\{3,4\}|A} & \text{if } x \oplus a_1 = 1 \end{cases}$$

   Bob sends $S_{3,2}$ *to Alice.*

5. *Alice executes private reverse cuts on the two pairs of $S_{3,2}$ using a_2 and a_3, respectively. Let the result be S_4.*

$$S_4 = \begin{cases} commit(0)^{\{1,2\},\{3,4\}|A} || commit(y)^{\{1,2\},\{3,4\}|A} \text{ if } x \oplus a_1 = 0 \\ commit(y)^{\{1,2\},\{3,4\}|A} || commit(0)^{\{1,2\},\{3,4\}|A} \text{ if } x \oplus a_1 = 1 \end{cases}$$

Alice then executes a private reverse selection on S_4 using a_1. Let S_5 be the result and the remaining two cards be S_6. The result $S_5 = commit(y)^{\{1,2\},\{3,4\}|A}$ if ($a_1 = 0$ and $x \oplus a_1 = 1$) or ($a_1 = 1$ and $x \oplus a_1 = 0$). The condition equals $x = 1$.
$S_5 = commit(0)^{\{1,2\},\{3,4\}|A}$ if ($a_1 = 0$ and $x \oplus a_1 = 0$) or ($a_1 = 1$ and $x \oplus a_1 = 1$). The condition equals $x = 0$. Thus,

$$S_5 = \begin{cases} commit(y)^{\{1,2\},\{3,4\}|A} \text{ if } x = 1 \\ commit(0)^{\{1,2\},\{3,4\}|A} \text{ if } x = 0 \end{cases}$$

$$= commit(x \wedge y)^{\{1,2\},\{3,4\}|A}$$

Alice sends S_5 and S_6 to Bob.
6. *Bob and Alice execute Protocol 2 (Base-fixed protocol) to $S_5||S_6$. Then they obtain $commit(x \wedge y)^{\{1,2\}}$.*

COPY Protocol

Protocol 4 *(Copy protocol) [17]*
Input: $commit(x)^{\{1,2\}}$ and two new cards 3 and 4.
Output: $commit(x)^{\{1,2\}}$ and $commit(x)^{\{3,4\}}$

1. *Alice executes a private random bisection cut on $commit(x)^{\{1,2\}}$. Let b be the random bit Alice selects. Alice sends the result, $commit(x \oplus b)^{\{1,2\}}$, to Bob.*
2. *Bob executes a private reveal on $commit(x \oplus b)^{\{1,2\}}$ and sees $x \oplus b$. Bob privately makes $commit(x \oplus b)^{\{3,4\}}$. Bob sends $commit(x \oplus b)^{\{1,2\}}$ and $commit(x \oplus b)^{\{3,4\}}$ to Alice.*
3. *Alice executes a private reverse cut on each of the pairs using b. The result is $commit(x)^{\{1,2\}}$ and $commit(x)^{\{3,4\}}$.*

The protocol is three rounds.

Preserving an Input In the above protocols to calculate Boolean functions, the input commitment values are lost. If the input is not lost, the input commitment can be used as an input to another calculation. Thus input preserving calculation is discussed [34, 37].

In the XOR protocol, $commit(y \oplus b)^{\{3,4\}}$ is no longer necessary after Bob sets the result. Thus, Bob can send back $commit(y \oplus b)^{\{3,4\}}$ to Alice. Then, Alice can recover $commit(y)^{\{3,4\}}$ using the private reverse cut. In this modified protocol, the output is $commit(x \oplus y)^{\{1,2\}}$ and $commit(y)^{\{3,4\}}$ without additional cards.

By exchanging the roles of x and y, the output can be $commit(x \oplus y)^{\{3,4\}}$ and $commit(x)^{\{1,2\}}$.

An input preserving AND protocol can be obtained using the idea in [34]. When we execute the AND protocol, two cards are selected by Alice at the final step. The remaining two cards are used to recover an input value. The unused two cards' value is

$$\begin{cases} 0 \text{ if } x = 1 \\ y \text{ if } x = 0 \end{cases}$$

thus the output is $commit(\overline{x} \wedge y)$.

Execute the above input preserving XOR protocol for these two output values so that the input $x \wedge y$ is preserved. The output of the XOR protocol is $(x \wedge y) \oplus (\overline{x} \wedge y) = y$. Thus, input y can be recovered without additional cards. By executing a base-fixed protocol, the output can be $commit(x \wedge y)^{\{1,2\}}$ and $commit(y)^{\{3,4\}}$.

***n*-input Boolean Functions** Since any 2-input Boolean function, NOT, and COPY can be executed, any n-input Boolean function can be calculated by the combination of the above protocols using $2n + 4$ cards by the idea in [34, 37].

Any Boolean function $f(x_1, x_2, \ldots, x_n)$ can be represented as follows: $f(x_1, x_2, \ldots, x_n) = \bar{x_1} \wedge \bar{x_2} \wedge \cdots \bar{x_n} \wedge f(0, 0, \ldots, 0) \oplus x_1 \wedge \bar{x_2} \wedge \cdots \bar{x_n} \wedge f(1, 0, \ldots, 0) \oplus \bar{x_1} \wedge x_2 \wedge \cdots \bar{x_n} \wedge f(0, 1, \ldots, 0) \oplus \cdots \oplus x_1 \wedge x_2 \wedge \cdots x_n \wedge f(1, 1, \ldots, 1)$.

Since the terms with $f(i_1, i_2, \ldots, i_n) = 0$ can be removed, this function f can be written as $f = \bigoplus_{i=1}^{k} v_1^i \wedge v_2^i \wedge \cdots \wedge v_n^i$, where $v_j^i = x_j$ or $\bar{x_j}$. Let us write $T_i = v_1^i \wedge v_2^i \wedge \cdots \wedge v_n^i$. The number of terms $k (< 2^n)$ depends on f.

Protocol 5 *(Protocol for any n-variable Boolean function [17]*

Input: $commit(x_i)^{\{2i+3,2i+4\}} (i = 1, 2, \ldots, n)$.

Output: $commit(f(x_1, x_2, \ldots, x_n))^{\{1,2\}}$.

The additional four cards (two pairs of cards) 1,2,3, and 4 are used as follows.

1 and 2 store the intermediate value to compute f.

3 and 4 store the intermediate value to compute T_i.

Execute the following steps for $i = 1, 2, \ldots, k$.

1. *Copy* v_1^i *from the input* $commit(x_1)$ *as* $commit(v_1^i)^{\{3,4\}}$. *(Note that if* v_1^i *is* $\bar{x_1}$, *NOT is taken after the copy).*
2. *For* $j = 2, \ldots, n$, *execute the following procedure: Execute the input preserving AND protocol to* $commit(\cdot)^{\{3,4\}}$ *and* $commit(v_j^i)$ *so that input* $commit(v_j^i)$ *is preserved. The result is stored as* $commit(\cdot)^{\{3,4\}}$. *(Note that if* v_j^i *is* $\bar{x_j}$, *NOT is taken before the AND protocol, and NOT is taken again for the preserved input.)*
   At the end of this step, T_i *is obtained as* $commit(v_1^i \wedge v_2^i \wedge \cdots \wedge v_n^i)^{\{3,4\}}$.
3. *If* $i = 1$, *copy* $commit(\cdot)^{\{3,4\}}$ *to* $commit(\cdot)^{\{1,2\}}$. *If* $i > 1$, *apply the XOR protocol between* $commit(\cdot)^{\{3,4\}}$ *and* $commit(\cdot)^{\{1,2\}}$. *The result is stored as* $commit(\cdot)^{\{1,2\}}$.

At the end of the protocol, $commit(f(x_1, x_2, \ldots x_n))^{\{1,2\}}$ *is obtained.*

3 Protocols for Three-input Boolean Functions

This section shows protocols for three input Boolean functions. The arguments to show the protocols with six cards are just the same as the one in [35]. The main difference is that logical AND can be calculated by four cards using private operations. In our protocols, no additional cards are necessary other than the cards for inputs.

There are $2^{2^3} = 256$ different functions with three inputs. However, some of these functions are equivalent by replacing variables and taking negations. NPN-classification [43] was considered to reduce the number of different functions considering the equivalence class of functions. The rules of NPN-classification are as follows.

1. Negation of input variables (Example: $x_i \leftrightarrow \overline{x_i}$).
2. Permutations of input variables (Example: $x_i \leftrightarrow x_j$).
3. Negation of the output ($f \leftrightarrow \overline{f}$).

For example, consider $f_1(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee x_3$. Several functions in the same equivalence class that includes f_1 are: $f_2 = (\overline{x_1} \wedge \overline{x_2}) \vee x_3$, $f_3 = (\overline{x_1} \wedge \overline{x_3}) \vee x_2$, $f_4 = \overline{f_3}$, and so on.

Input negation and output negation can be executed by card-based protocols without increasing the number of cards. They are executed by just swapping input cards or output cards. Permutations of input variables can also be executed without increasing the number of cards. They can be achieved by just changing the positions of the input values. Therefore, all functions in the same NPN equivalence class can be calculated with the same number of cards.

Theorem 1. *Any three input Boolean functions can be securely calculated without additional cards other than the input cards with a standard deck of cards when we use private operations.*

Proof. When the number of inputs is 3, there are the following 14 NPN-representative functions [43]. (Note that x, y, and z are used to represent input variables.)

1. $NPN_1 = 1$
2. $NPN_2 = x$
3. $NPN_3 = x \vee y$
4. $NPN_4 = x \oplus y$
5. $NPN_5 = x \wedge y \wedge z$
6. $NPN_6 = (x \wedge y \wedge z) \vee (\overline{x} \wedge \overline{y} \wedge \overline{z})$
7. $NPN_7 = (x \wedge y) \vee (x \wedge z)$
8. $NPN_8 = (x \wedge y) \vee (\overline{x} \wedge \overline{y} \wedge z)$
9. $NPN_9 = (x \wedge y \wedge \overline{z}) \vee (x \wedge \overline{y} \wedge z) \vee (\overline{x} \wedge y \wedge z)$
10. $NPN_{10} = (x \wedge \overline{y} \wedge \overline{z}) \vee (\overline{x} \wedge y \wedge \overline{z}) \vee (\overline{x} \wedge \overline{y} \wedge z) \vee (x \wedge y \wedge z) = x \oplus y \oplus z$.
11. $NPN_{11} = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$
12. $NPN_{12} = (x \wedge \overline{z}) \vee (y \wedge z)$
13. $NPN_{13} = (x \wedge y \wedge z) \vee (x \wedge \overline{y} \wedge \overline{z})$
14. $NPN_{14} = (x \wedge y) \vee (x \wedge z) \vee (\overline{x} \wedge \overline{y} \wedge \overline{z})$

Among these 14 functions, NPN_1 - NPN_4 depend on less than three inputs. These functions can be calculated without additional cards [17]. We show a calculation protocol for each of the remaining functions. Note that the output is $commit(f)^{\{1,2\}}$ when the inputs are $commit(x)^{\{1,2\}}$, $commit(y)^{\{3,4\}}$, and $commit(z)^{\{5,6\}}$.

For NPN_5, $x \wedge y$ can be calculated without additional cards. Then $x \wedge y \wedge z$ can be calculated without additional cards other than the input cards, $x \wedge y$ and z.

NPN_7 can be represented as $NPN_7 = x \wedge (y \vee z)$, thus this function can also be calculated without additional cards.

NPN_{10} can be calculated as $(x \oplus y) \oplus z$ without additional cards.

NPN_{13} can be represented as $NPN_{13} = x \wedge (\overline{y \oplus z})$, thus this function can also be calculated without additional cards.

NPN_{14} can be represented as $NPN_{14} = \overline{x} \oplus (y \vee z)$, thus this function can also be calculated without additional cards.

NPN_6 can be represented as $NPN_6 = (\overline{x \oplus y}) \wedge (\overline{x \oplus z})$. First, calculate $commit(x \oplus y)^{\{3,4\}}$ with preserving input $commit(x)^{\{1,2\}}$. Then calculate $commit(x \oplus z)^{\{1,2\}}$. Then NOT is applied to each result. Next, calculate AND to these results.

NPN_8 can be represented as $NPN_8 = (\overline{x \oplus y}) \wedge (y \vee z)$. First, calculate $commit(x \oplus y)^{\{1,2\}}$ with preserving input $commit(y)^{\{3,4\}}$. Then NOT is applied to the result. Then calculate $commit(y \vee z)^{\{3,4\}}$. Next, calculate AND to these results.

NPN_9 can be represented as $NPN_9 = (\overline{x \oplus y \oplus z}) \wedge (x \vee z)$. First, calculate $commit(x \oplus y)^{\{3,4\}}$ with preserving input $commit(x)^{\{1,2\}}$. Next, calculate $commit((x \oplus y) \oplus z)^{\{3,4\}}$ with preserving $commit(z)^{\{5,6\}}$. Then NOT is applied to the result. Next, calculate $commit(x \vee z)^{\{1,2\}}$. Next, calculate AND to these results.

NPN_{12} can be calculated as follows. First, calculate $commit(x \wedge \overline{z})^{\{1,2\}}$ with preserving input $commit(z)^{\{5,6\}}$. Next, calculate $commit(y \wedge z)^{\{3,4\}}$. Then, calculate OR to these results by using the AND protocol.

NPN_{11} can be represented as

$$NPN_{11} = \begin{array}{ll} z & \text{if } x \oplus y = 1 \\ x & \text{if } x \oplus y = 0 \end{array}$$

Since this equation is similar to the AND equation, the function can be calculated by modifying the AND protocol as follows.

1. Alice and Bob calculate $commit(x \oplus y)^{\{3,4\}}$ with preserving input $commit(x)^{\{1,2\}}$.
2. Alice executes private random bisection cut on $commit(x \oplus y)^{\{3,4\}}$, $commit(x)^{\{1,2\}}$, and $commit(z)^{\{5,6\}}$ using different random bit $a_1, a_2, a_3 \in \{0, 1\}$. Alice sends the result $commit(x \oplus y \oplus a_1)^{\{3,4\}}$, $commit(x \oplus a_2)^{\{1,2\}}$, and $commit(z \oplus a_3)^{\{5,6\}}$ to Bob.
3. Bob privately select a bit $b_1 \in \{0, 1\}$ and exchanges the bases of $commit(x \oplus a_2)^{\{1,2\}}$ and $commit(z \oplus a_3)^{\{5,6\}}$ if $b_1 = 1$. Though Bob sees the committed values, Bob obtains no information about x and z since Alice randomized

the values. Bob sends $commit(x \oplus a_2)^{\{1,2\},\{5,6\}|A} || commit(z \oplus a_3)^{\{1,2\},\{5,6\}|A}$ to Alice.
4. Alice executes private reverse cuts to the sequence using a_2 and a_3. Alice sends the result $commit(x)^{\{1,2\},\{5,6\}|A} || commit(z)^{\{1,2\},\{5,6\}|A}$ to Bob.
5. Bob executes private reveal on $commit(x \oplus y \oplus a_1)$. Bob sets

$$S_2 = \begin{cases} commit(z)^{\{1,2\},\{5,6\}|A} || commit(x)^{\{1,2\},\{5,6\}|A} \text{ if } x \oplus y \oplus a_1 = 1 \\ commit(x)^{\{1,2\},\{5,6\}|A} || commit(z)^{\{1,2\},\{5,6\}|A} \text{ if } x \oplus y \oplus a_1 = 0 \end{cases}$$

6. Alice executes a private reverse cut on S_2 using the bit a_1 generated in the private random bisection cut. Let the obtained sequence be S_3. S_3 is $commit(z)^{\{1,2\},\{5,6\}|A} || commit(x)^{\{1,2\},\{5,6\}|A}$ if $(x \oplus y \oplus a_1 = 1$ and $a_1 = 0)$ or $(x \oplus y \oplus a_1 = 0$ and $a_1 = 1)$. The case equals to $x \oplus y = 1$. The output is $commit(x)^{\{1,2\},\{5,6\}|A} || commit(z)^{\{1,2\},\{5,6\}|A}$ if $(x \oplus y \oplus a_1 = 1$ and $a_1 = 1)$ or $(x \oplus y \oplus a_1 = 0$ and $a_1 = 0)$. The case equals to $x \oplus y = 0$. Thus the result is

$$S_3 = \begin{cases} commit(z)^{\{1,2\},\{5,6\}|A} || commit(x)^{\{1,2\},\{5,6\}|A} \text{ if } x \oplus y = 1 \\ commit(x)^{\{1,2\},\{5,6\}|A} || commit(z)^{\{1,2\},\{5,6\}|A} \text{ if } x \oplus y = 0 \end{cases}$$

Note that the left pair has the value of the result.
7. Alice and Bob execute base-fixed protocol on S_3. They obtain

$$\begin{cases} commit(z)^{\{1,2\}} \text{ if } x \oplus y = 1 \\ commit(x)^{\{1,2\}} \text{ if } x \oplus y = 0 \end{cases}$$

Therefore, NPN_{11} can also be calculated without additional cards. □

4 Half Adder, Full Adder, and Symmetric Functions

This section first shows a realization of half adder and full adder. In our protocols, no additional cards are necessary other than the cards for inputs.

The input and output of the secure half adder are as follows:

- Input: $commit(x)^{\{1,2\}}$ and $commit(y)^{\{3,4\}}$
- Output: $S = commit(x \oplus y)^{\{3,4\}}$ and $C = commit(x \wedge y)^{\{1,2\}}$

The half adder is realized by the following steps, whose idea is just the same as the one in [34].

1. Execute XOR protocol with preserving input x. Thus $commit(x)^{\{1,2\}}$ and $commit(x \oplus y)^{\{3,4\}}$ are obtained.
2. Obtain $commit(\overline{x \oplus y})^{\{3,4\}}$ by swapping the two cards of $commot(x \oplus y)^{\{3,4\}}$.
3. Execute AND protocol to $commit(x)^{\{1,2\}}$ and $commit(\overline{x \oplus y})^{\{3,4\}}$ with preserving input $commit(\overline{x \oplus y})^{\{3,4\}}$. Thus $commit(\overline{x \oplus y})^{\{3,4\}}$ and $commit(x \wedge \overline{(x \oplus y)})^{\{1,2\}} = commit(x \wedge y)^{\{1,2\}}$ are obtained.
4. Obtain $commit(x \oplus y)^{\{1,2\}}$ by swapping the two cards of $commit(\overline{x \oplus y})^{\{1,2\}}$.

No additional cards are necessary other than the four input cards.

The input and output of the secure full adder are as follows:

- Input: $commit(C_I)^{\{1,2\}}$, $commit(x)^{\{3,4\}}$, and $commit(y)^{\{5,6\}}$.
- Output: $C_O = commit((x \wedge y) \vee (x \wedge C_I) \vee (y \wedge C_I))^{\{1,2\}}$ and $S = commit(x \oplus y \oplus C_I)^{\{3,4\}}$.

Since the half adder can be calculated without additional cards, the full adder can also be calculated without additional cards by the following protocol.

1. Add C_I and x using the half adder. The outputs are $commit(x \oplus c_I)^{\{3,4\}}$ and $commit(x \wedge C_I)^{\{1,2\}}$.
2. Add $commit(y)^{\{5,6\}}$ to the result $commit(x \oplus C_I)^{\{3,4\}}$ using the half adder. The outputs are $commit(x \oplus y \oplus C_I)^{\{3,4\}}$ and $commit(y \wedge (x \oplus C_I))^{\{5,6\}}$.
3. Execute OR protocol to $commit(y \wedge (x \oplus C_I))^{\{5,6\}}$ and $commit(x \wedge C_I)^{\{1,2\}}$. Since $(y \wedge (x \oplus C_I)) \vee (x \wedge C_I) = (x \wedge y) \vee (x \wedge C_I) \vee (y \wedge C_I)$, the carry C_O is obtained by the base of $\{1, 2\}$.

Using the half adder and full adder, calculation of symmetric function can be done by the technique in [34]. n-input symmetric function $f(x_1, x_2, \ldots, x_n)$ depends only on the number of variables such that $x_i = 1$. Let $Y = \sum_{i=1}^{n} x_i$. Then the function f can be written as $f(x_1, x_2, \ldots, x_n) = g(Y)$. When Y is given by a binary representation, $Y = y_k y_{k-1} ... y_1$, g can be written as $g(y_1, y_2, \ldots, y_k)$, where $k = \lfloor \log n \rfloor + 1$.

Given input $x_1, x_2, \ldots, x_n$, first, obtain the sum of these inputs using the half adder and full adder protocols without additional cards. The sum is obtained as $y_1, y_2, \ldots, y_k$. Then, calculate g using y_i. When $n \leq 7$, $k \leq 3$, thus any three input Boolean function g can be calculated without additional cards. When $n \geq 8$, Y is represented with $k = \lfloor \log n \rfloor + 1$ bits. Since $n - k \geq 4$, at least 8 input cards are unused after y_is are calculated. Any Boolean function can be calculated with four additional cards, thus g can be calculated without additional cards other than the input cards.

Theorem 2. *Any symmetric Boolean function can be securely calculated without additional cards other than the input cards when we use private operations.*

5 Conclusion

This paper showed card-based cryptographic protocols to calculate three input Boolean functions, half adder, full adder, and symmetric functions with a standard deck of cards using private operations. These results show the effectiveness of private operations.

One of the important open problems is obtaining another class of Boolean functions that can be calculated without additional cards using private operations. However, it seems very difficult to achieve all four-input Boolean functions without additional cards.

References

1. Abe, Y., Hayashi, Y.i., Mizuki, T., Sone, H.: Five-card and computations in committed format using only uniform cyclic shuffles. New Generation Computing **39**(1), 97–114 (2021)
2. Abe, Y., Mizuki, T., Sone, H.: Committed-format and protocol using only random cuts. Natural Computing pp. 1–7 (2021)
3. den Boer, B.: More efficient match-making and satisfiability the five card trick. In: Proc. of EUROCRYPT '89, LNCS Vol. 434. pp. 208–217 (1990)
4. Cheung, E., Hawthorne, C., Lee, P.: Cs 758 project: Secure computation with playing cards (2013), `http://cdchawthorne.com/writings/secure\_playing\_cards.pdf`
5. Francis, D., Aljunid, S.R., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Necessary and sufficient numbers of cards for securely computing two-bit output functions. In: Proc. of Second International Conference on Cryptology and Malicious Security(Mycrypt 2016), LNCS Vol. 10311. pp. 193–211 (2017)
6. Hanaoka, G., Iwamoto, M., Watanabe, Y., Mizuki, T., Abe, Y., Shinagawa, K., Arai, M., Yanai, N.: Physical and visual cryptography to accelerate social implementation of advanced cryptographic technologies. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences pp. 214–228 (2023), (In Japanese)
7. Isuzugawa, R., Toyoda, K., Sasaki, Y., Miyahara, D., Mizuki, T.: A card-minimal three-input and protocol using two shuffles. In: Proc. of 27th International Computing and Combinatorics Conference (COCOON 2021), LNCS Vol. 13025. pp. 668–679. Springer (2021)
8. Kastner, J., Koch, A., Walzer, S., Miyahara, D., Hayashi, Y., Mizuki, T., Sone, H.: The minimum number of cards in practical card-based protocols. In: Proc. of Asiacrypt 2017, Part III, LNCS Vol. 10626. pp. 126–155 (2017)
9. Koch, A., Schrempp, M., Kirsten, M.: Card-based cryptography meets formal verification. New Generation Computing **39**(1), 115–158 (2021)
10. Koch, A., Walzer, S.: Foundations for actively secure card-based cryptography. In: Proc. of 10th International Conference on Fun with Algorithms (FUN 2020). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
11. Koch, A., Walzer, S., Härtel, K.: Card-based cryptographic protocols using a minimal number of cards. In: Proc. of Asiacrypt 2015, LNCS Vol. 9452. pp. 783–807 (2015)
12. Koyama, H., Miyahara, D., Mizuki, T., Sone, H.: A secure three-input and protocol with a standard deck of minimal cards. In: Santhanam, R., Musatov, D. (eds.) Proc. of 16th International Computer Science Symposium in Russia (CSR 2021), LNCS Vol. 12730. pp. 242–256. Springer International Publishing, Cham (2021)
13. Koyama, H., Toyoda, K., Miyahara, D., Mizuki, T.: New card-based copy protocols using only random cuts. In: Proceedings of the 8th ACM on ASIA Public-Key Cryptography Workshop. pp. 13–22. APKC '21, Association for Computing Machinery, New York, NY, USA (2021)
14. Kuzuma, T., Isuzugawa, R., Toyoda, K., Miyahara, D., Mizuki, T.: Card-based single-shuffle protocols for secure multiple-input and and xor computations. In: Proceedings of the 9th ACM on ASIA Public-Key Cryptography Workshop. pp. 51–58 (2022)
15. Manabe, Y., Ono, H.: Card-based cryptographic protocols for three-input functions using private operations. In: Proc. of 32nd International Workshop on Combinatorial Algorithms (IWOCA 2021), LNCS Vol. 12757. pp. 469–484. Springer (2021)

16. Manabe, Y., Ono, H.: Card-based cryptographic protocols with malicious players using private operations. New Generation Computing **40**(1), 67–93 (2022)
17. Manabe, Y., Ono, H.: Card-based cryptographic protocols with a standard deck of cards using private operations. New Generation Computing (2024)
18. Marcedone, A., Wen, Z., Shi, E.: Secure dating with four or fewer cards. IACR Cryptology ePrint Archive, Report 2015/1031 (2015)
19. Miyahara, D., Hayashi, Y.i., Mizuki, T., Sone, H.: Practical card-based implementations of yao's millionaire protocol. Theoretical Computer Science **803**, 207–221 (2020)
20. Miyahara, D., Mizuki, T.: Secure computations through checking suits of playing cards. In: Proc. of International Workshop on Frontiers in Algorithmic Wisdom (IJTCS-FAW 2022), LNCS Vol. 13461. pp. 110–128. Springer (2022)
21. Mizuki, T.: Applications of card-based cryptography to education. In: IEICE Technical Report ISEC2016-53. pp. 13–17 (2016), (In Japanese)
22. Mizuki, T.: Card-based protocols for securely computing the conjunction of multiple variables. Theoretical Computer Science **622**, 34–44 (2016)
23. Mizuki, T.: Efficient and secure multiparty computations using a standard deck of playing cards. In: Proc. of 15th International Conference on Cryptology and Network Security(CANS 2016), LNCS Vol.10052. pp. 484–499. Springer (2016)
24. Mizuki, T., Komano, Y.: Information leakage due to operative errors in card-based protocols. Information and Computation **285**, 104910 (2022)
25. Mizuki, T., Kumamoto, M., Sone, H.: The five-card trick can be done with four cards. In: Proc. of Asiacrypt 2012, LNCS Vol.7658. pp. 598–606 (2012)
26. Mizuki, T., Shizuya, H.: A formalization of card-based cryptographic protocols via abstract machine. International Journal of Information Security **13**(1), 15–23 (2014)
27. Mizuki, T., Shizuya, H.: Practical card-based cryptography. In: Proc. of 7th International Conference on Fun with Algorithms(FUN2014), LNCS Vol. 8496. pp. 313–324 (2014)
28. Mizuki, T., Shizuya, H.: Computational model of card-based cryptographic protocols and its applications. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **100**(1), 3–11 (2017)
29. Mizuki, T., Sone, H.: Six-card secure and and four-card secure xor. In: Proc. of 3rd International Workshop on Frontiers in Algorithms(FAW 2009), LNCS Vol. 5598. pp. 358–369 (2009)
30. Morooka, T., Manabe, Y., Shinagawa, K.: Malicious player card-based cryptographic protocols with a standard deck of cards using private operations. In: Proc. of 18th International Conference on Information Security Practice and Experience (ISPEC 2023), LNCS vol. 14341. pp. 332–346. Springer Nature Singapore (2023)
31. Nakai, T., Iwanari, K., Ono, T., Abe, Y., Watanabe, Y., Iwamoto, M.: Card-based cryptography with a standard deck of cards, revisited: Efficient protocols in the private model. New Generation Computing (2024)
32. Nakai, T., Misawa, Y., Tokushige, Y., Iwamoto, M., Ohta, K.: How to solve millionaires' problem with two kinds of cards. New Generation Computing **39**(1), 73–96 (2021)
33. Niemi, V., Renvall, A.: Solitaire zero-knowledge. Fundamenta Informaticae **38**(1, 2), 181–188 (1999)
34. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols for any boolean function. In: Proc. of 15th International Conference on Theory and Applications of Models of Computation(TAMC 2015), LNCS Vol. 9076. pp. 110–121 (2015)

35. Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Securely computing three-input functions with eight cards. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **98**(6), 1145–1152 (2015)
36. Nishimura, A., Nishida, T., Hayashi, Y., Mizuki, T., Sone, H.: Card-based protocols using unequal division shuffles. Soft Computing **22**(2), 361–371 (2018)
37. Ono, H., Manabe, Y.: Card-based cryptographic logical computations using private operations. New Generation Computing **39**(1), 19–40 (2021)
38. Ono, H., Manabe, Y.: Minimum round card-based cryptographic protocols using private operations. Cryptography **5**(3) (2021)
39. Ruangwises, S.: Two standard decks of playing cards are sufficient for a zkp for sudoku. New Generation Computing **40**(1), 49–65 (2022)
40. Ruangwises, S., Itoh, T.: And protocols using only uniform shuffles. In: Proc. of 14th International Computer Science Symposium in Russia(CSR 2019), LNCS Vol. 11532. pp. 349–358 (2019)
41. Ruangwises, S., Itoh, T.: Securely computing the n-variable equality function with 2n cards. Theoretical Computer Science **887**, 99–110 (2021)
42. Ruangwises, S., Itoh, T.: Physical zkp for makaro using a standard deck of cards. In: Proc. of 17th International Conference on Theory and Applications of Models of Computation (TAMC 2022), LNCS Vol. 13571. pp. 43–54. Springer (2022)
43. Sasao, T., Butler, J.T.: Progress in Applications of Boolean Functions. Morgan and Claypool Publishers (2010)
44. Shikata, H., Miyahara, D., Mizuki, T.: Few-helping-card protocols for some wider class of symmetric boolean functions with arbitrary ranges. In: Proceedings of the 10th ACM International Workshop on ASIA Public-Key Cryptography (APKC). pp. 33–41 (2023)
45. Shikata, H., Toyoda, K., Miyahara, D., Mizuki, T.: Card-minimal protocols for symmetric boolean functions of more than seven inputs. In: Seidl, H., Liu, Z., Pasareanu, C.S. (eds.) Proc. of 18th International Conference on Theoretical Aspects of Computing (ICTAC 2022) LNCS Vol. 13572. pp. 388–406. Springer International Publishing, Cham (2022)
46. Shinagawa, K., Mizuki, T.: The six-card trick:secure computation of three-input equality. In: Proc. of 21st International Conference on Information Security and Cryptology (ICISC 2018), LNCS Vol. 11396. pp. 123–131 (2018)
47. Shinagawa, K., Mizuki, T.: Secure computation of any boolean function based on any deck of cards. In: Proc. of 13th International Workshop on Frontiers in Algorithmics (FAW 2019), LNCS Vol. 11458. pp. 63–75. Springer (2019)
48. Shinagawa, K., Nuida, K.: A single shuffle is enough for secure card-based computation of any boolean circuit. Discrete Applied Mathematics **289**, 248–261 (2021)
49. Takashima, K., Miyahara, D., Mizuki, T., Sone, H.: Actively revealing card attack on card-based protocols. Natural Computing **21**(4), 615–628 (2022)
50. Toyoda, K., Miyahara, D., Mizuki, T., Sone, H.: Six-card finite-runtime xor protocol with only random cut. In: Proc. of the 7th ACM Workshop on ASIA Public-Key Cryptography. pp. 2–8 (2020)
51. Yoshida, T., Tanaka, K., Nakabayashi, K., Chida, E., Mizuki, T.: Upper bounds on the number of shuffles for two-helping-card multi-input and protocols. In: Proc. of 22nd International Conference on Cryptology and Network Security(CANS 2023), LNCS vol. 14342. pp. 211–231. Springer (2023)

Grid-Based Decompositions for Spatial Data under Local Differential Privacy

Berkay Kemal Balioglu, Alireza Khodaie, Ameer Taweel, and M. Emre Gursoy

Department of Computer Engineering, Koç University, Istanbul, Turkey
{bbalioglu23, akhodaie22, ataweel20, emregursoy}@ku.edu.tr

Abstract. Local differential privacy (LDP) has recently emerged as a popular privacy standard. With the growing popularity of LDP, recent works have applied LDP to spatial data, and grid-based decompositions have been a common building block in DP and LDP. In this paper, we study three grid-based decomposition methods for spatial data under LDP: Uniform Grid (UG), PrivAG, and AAG. UG is a static approach that consists of equal-sized cells. To enable data-dependent decomposition, PrivAG was proposed by Yang et al. (2022). To advance the state-of-the-art in adaptive grids, this paper proposes the Advanced Adaptive Grid (AAG) method. For each grid cell, following the intuition that the cell's intra-cell density distribution will be affected by its neighbors, AAG performs uneven cell divisions depending on the neighboring cells' densities. We experimentally compare UG, PrivAG, and AAG using three real-world location datasets, varying privacy budgets, and query sizes. Results show that AAG provides higher utility than PrivAG, demonstrating the superiority of our proposed approach. Furthermore, when the grid size is chosen optimally in UG, AAG still beats UG for small queries, but UG beats AAG for large (coarse-grained) queries.

Keywords: Local differential privacy · location privacy · spatial grids · location-based services · spatial data management.

1 Introduction

Large volumes of spatial data are nowadays available for collection and analysis, thanks to the popularity of smartphones, connected cars, location-based services (LBS), and social networks. Ensuring the privacy of spatial data is imperative since it contains sensitive information about individuals, such as their home and work addresses, frequently visited locations, and personal habits. Hence, users are reluctant to share their location data with untrusted data collectors. In recent years, local differential privacy (LDP) has become a widely accepted standard for privacy protection and deployed in products of various companies such as Apple, Google, and Microsoft [2,4,6,7]. With the growing popularity of LDP, several recent works have applied LDP to spatial data and LBS [1,5,9,11,13,14]. However, in many applications of DP and LDP to spatial data, the data needs to be discretized so that the input and output domains of the privacy mechanisms

are discrete and finite. Grid-based decompositions, which decompose the overall geospatial area Ω into non-overlapping cells, are a popular method for this purpose. After a grid is laid, the user's location can be discretized by determining which cell it falls inside. Indeed, uniform and adaptive grids have been widely used in the DP and LDP literature for trajectory collection and sharing [5,13], range query answering [10], synthetic data generation [8], and so forth.

A uniform grid (UG) partitions the geospatial area Ω into $N \times M$ cells of equal size. However, since the uniform grid does not adapt to the underlying data distribution [10,13], it may result in a poor partitioning when certain regions of Ω have too high or too low density. To enable density-dependent decomposition of Ω, the adaptive grid approach was proposed [10,13,8]. Its main idea is to first lay a uniform grid $\mathcal{G}_1$ over the given Ω, and according to the cell density estimations obtained from a portion of the users, further divide each individual cell (i.e., adapt the grid). To the best of our knowledge, the most recent adaptive grid approach in LDP is PrivAG by Yang et al. [13].

In this paper, we propose Advanced Adaptive Grid (AAG). In PrivAG, when a certain cell $C_k \in \mathcal{G}_1$ needs to be divided further, this division is done evenly. In AAG, we propose to perform this division by taking into account C_k's neighbor cells because C_k's intra-cell density distribution is likely to mimic its neighbors' densities. For example, if C_k's right neighbor is dense but the left neighbor is sparse, then the intra-cell density of C_k is likely to be skewed towards the right. Following this intuition, we perform an uneven division which is weighted proportional to the neighbors' densities. Furthermore, we propose heuristic strategies to handle edge cells and corner cells that lack one or more neighbors. In addition, motivated by our observation that the parameter choices in PrivAG yield cell counts too similar to $\mathcal{G}_1$, we propose new parameter values for AAG.

We experimentally compare UG, PrivAG, and AAG using three real-world location datasets (Gowalla, Porto, Foursquare) by measuring their Average Query Errors (AQE) in answering spatial density queries with different privacy budgets ε and query sizes ρ. We find that the AQEs of UG are heavily dependent on the grid size, i.e., it performs well when the grid size is chosen optimally, but poorly otherwise. Comparing UG with optimal grid sizes against PrivAG and AAG, we observe that: (i) AAG is preferable to PrivAG across all ε and ρ, demonstrating that AAG improves the state-of-the-art in adaptive grids in LDP, (ii) AAG is the best approach when ρ is small but UG is the best approach when ρ is large.

2 Background

Let $\mathcal{U} = \{u_1, u_2, u_3, ...\}$ denote the set of users where $|\mathcal{U}|$ is the total number of users, and let the two-dimensional geospatial area be denoted by Ω. For each user u_i, the user's true location is represented by l_i, such that $l_i \in \Omega$ and l_i consists of a pair of *(latitude, longitude)* coordinates. We assume that the boundaries of the overall domain Ω are not privacy-sensitive, and can be known by all parties. Yet, each user's location is privacy-sensitive and must be protected.

Local differential privacy (LDP) is a widely accepted standard for safeguarding privacy. In LDP, users' data is perturbed on their devices before being col-

lected by the aggregator (also called the "server"). After collecting perturbed data, the server uses estimation methods to recover statistics pertaining to the general population. However, since each user's data is perturbed, the server cannot infer exact information about a specific user. In our context, since each user's location l_i needs to be protected, we define LDP as follows.

Definition 1 (ε-LDP). A randomized algorithm Ψ satisfies ε-local differential privacy (ε-LDP), where $\varepsilon > 0$, if and only if for any two inputs l_i, l_i^*:

$$\forall y \in Range(\Psi): \quad \frac{Pr[\Psi(l_i) = y]}{Pr[\Psi(l_i^*) = y]} \leq e^{\varepsilon} \tag{1}$$

where $Range(\Psi)$ stands for the set of all possible outputs of the algorithm Ψ.

ε-LDP ensures that having observed the output y, the server (or any other party who observed y) is not able to distinguish whether the original location of the user was l_i or l_i^* with probability more than the odds ratio controlled by e^{ε}. The strength of the privacy protection is controlled by the parameter ε, commonly known as the *privacy budget*. Lower ε yields stronger privacy.

Numerous LDP protocols have been proposed to minimize utility loss and/or communication cost under various conditions. In this paper, we use a state-of-the-art protocol called *Optimized Local Hashing (OLH)* [12] due to its high utility and low communication cost [3,7]. Similar to other LDP protocols, OLH consists of two main components: (i) user-side encoding and perturbation on users' devices, and (ii) server-side estimation after collecting perturbed data from the user population. Due to the page limit, we refer the reader to [12] for the components' technical descriptions. Note that although we use OLH as the LDP protocol, the grid methods are not specific to OLH. Other protocols such as GRR, RAPPOR, OUE, and the Staircase Mechanism can also be used [11,12].

3 Grid-Based Decompositions Under LDP

We first describe the Uniform Grid (UG) approach in Section 3.1, then the existing adaptive grid approach called PrivAG [13] in Section 3.2, and finally our novel Advanced Adaptive Grid approach called AAG in Section 3.3.

3.1 Uniform Grid (UG)

A uniform grid partitions the geospatial area Ω into $N \times M$ cells of equal size. We denote this grid by $\mathcal{G}_{uni} = (C_1, C_2, ..., C_{N \times M})$ where each $C_j \in \mathcal{G}$ is one cell. The geographic coverage of all cells are disjoint from one another. User u_i with location l_i discretizes his/her location by finding which $C_i \in \mathcal{G}$ their l_i falls inside. Then, to satisfy LDP, the cell information C_i needs to be perturbed. Hence, we feed C_i into the OLH protocol with the appropriate parameters.

An overview of LDP location data collection using $\mathcal{G}_{uni}$ is shown in Algorithm 1. For each user u_i with location l_i, the user first finds their true cell C_i, i.e., which

Algorithm 1 Collecting location data with LDP using a grid

1: **Input:** Users $\mathcal{U}$, grid $\mathcal{G}$, privacy budget ε
2: **Output:** Densities of each cell in $\mathcal{G}$
3:
4: ▷ **User-side discretization and perturbation**
5: **for** each user $u_i \in \mathcal{U}$ **do**
6: **for** each cell $C_j \in \mathcal{G}$ **do**
7: **if** l_i falls inside C_j **then**
8: Set user's true cell as: $C_i \leftarrow C_j$
9: **break**
10: Execute user-side OLH with true value = C_i, domain = $\mathcal{G}$, and budget = ε
11: Send the resulting tuple $\langle H_u, x'_u \rangle$ to the server
12:
13: ▷ **Server-side estimation**
14: Server receives $\langle H_u, x'_u \rangle$ from all $u_i \in \mathcal{U}$
15: **for** each cell $C_j \in \mathcal{G}$ **do**
16: Compute $Sup(C_j)$ as the number of tuples for which $x'_u = H_u(C_j)$
17: Compute $\Phi(C_j)$ using the server-side estimation of OLH
18: **return** $\Phi(C_1)$, $\Phi(C_2)$, ... for all $C_j \in \mathcal{G}$

cell in the grid their location falls inside (lines 6-9). Then, the user executes the OLH protocol by treating their true value as C_i, the domain of the protocol as $\mathcal{G}_{uni} = (C_1, C_2, ..., C_{N \times M})$, and using the privacy budget ε. Since l_i is discretized as C_i, the domain of OLH is also discretized: $\mathcal{D} = \mathcal{G}_{uni} = (C_1, C_2, ..., C_{N \times M})$, instead of using a continuous domain $\mathcal{D} = \Omega$. Each user sends the OLH protocol output to the server. The server receives the outputs from all users and then estimates the density of each cell $C_j \in \mathcal{G}_{uni}$ (lines 15-17) using the server-side estimation procedure of OLH.

3.2 Existing Adaptive Grid: PrivAG

UG is a static approach that does not adapt to the underlying data distribution [10,13]. It may result in a poor partitioning of Ω when certain regions have too high or too low density. For example, when a cell is too crowded, then further partitioning it into smaller cells enables a better understanding of the detailed data distribution *within* that cell. Yet, the uniform grid is not able to achieve this. On the other hand, if a certain region of Ω is sparse, then the cells in that region will have zero or near-zero density, and the uniform grid will be over-partitioning that region. Over-partitioned cells lead to utility loss since their estimated densities are non-zero due to LDP perturbation, leading to fictitious and skewed results. To enable data-dependent decompositions, the adaptive grid approach was proposed in DP and LDP [10,13]. Its main idea is to first lay a uniform grid $\mathcal{G}_1$ over the given Ω, and according to the cell density estimations obtained from users, further divide each individual cell (i.e., adapt the grid).

To the best of our knowledge, the most recent adaptive grid approach in LDP is PrivAG, proposed by Yang et al. [13]. An algorithmic overview of PrivAG is

Algorithm 2 Algorithmic summary of the PrivAG approach

Input: Users $\mathcal{U}$, parameters α and σ, privacy budget ε
Output: Densities of each cell in adaptive grid $\mathcal{G}_{ag}$

▷ **First phase of PrivAG**
Server computes g_1 and divides $\mathcal{U}$ into $\mathcal{U}_1$ and $\mathcal{U}_2$ such that $|\mathcal{U}_1| = \sigma \times |\mathcal{U}|$
Server lays $g_1 \times g_1$ uniform grid $\mathcal{G}_1$ on Ω
Call Algorithm 1 with $\mathcal{U}_1$, $\mathcal{G}_1$ and ε to obtain $\Phi(C_1)$, $\Phi(C_2)$, ... for all $C_k \in \mathcal{G}_1$

▷ **Second phase of PrivAG**
for each cell $C_k \in \mathcal{G}_1$ **do**
    Compute g_2^k and divide C_k into $g_2^k \times g_2^k$ cells of equal size
Let $\mathcal{G}_{ag}$ denote the resulting grid after the above divisions
Call Algorithm 1 with $\mathcal{U}_2$, $\mathcal{G}_{ag}$ and ε to obtain $\Phi(C_1)$, $\Phi(C_2)$, ... for all cells in $\mathcal{G}_{ag}$

given in Algorithm 2. In PrivAG, the server first divides the set of users $\mathcal{U}$ into two groups: $\mathcal{U}_1$ and $\mathcal{U}_2$. Then, the server constructs a uniform grid $\mathcal{G}_1$ of size $g_1 \times g_1$ and broadcasts $\mathcal{G}_1$ to users in $\mathcal{U}_1$. Based on $\mathcal{G}_1$, users in $\mathcal{U}_1$ discretize their locations and use OLH to send their perturbed outcomes to the server. The server estimates the densities of each cell in $\mathcal{G}_1$. Then, for each cell $C_k \in \mathcal{G}_1$, the server further partitions C_k into $g_2^k \times g_2^k$ cells, where g_2^k depends on $\Phi(C_k)$ and other parameters. After all cells are partitioned according to their g_2^k, the resulting adaptive grid $\mathcal{G}_{ag}$ is obtained. Then, $\mathcal{G}_{ag}$ is advertised to users in $\mathcal{U}_2$ and desired statistics are obtained using $\mathcal{G}_{ag}$, e.g., cell densities. Since the values of g_1 and g_2^k have an important impact on the final grid, Yang et al. propose guidelines for choosing them in [13].

3.3 Proposed Approach: Advanced Adaptive Grid (AAG)

We propose the Advanced Adaptive Grid (AAG) approach which advances PrivAG. Consider Figure 1, which exemplifies a uniform grid with user densities written inside the cells (on the left), PrivAG (in the middle), and AAG (on the right). Say that the first phase of PrivAG laid a 3×3 uniform grid $\mathcal{G}_1$ on Ω, and the resulting densities of cells are shown on the left of Figure 1. In its second phase, PrivAG iterates through each $C_k \in \mathcal{G}_1$ and decides how to further divide C_k. Say that in the current iteration, C_k is the middle cell in Figure 1, and it is found that $g_2 = 2$. Then, PrivAG divides the middle cell into $2 \times 2 = 4$ equally sized cells. Our intuition is that the division of this middle cell into equal-sized cells is suboptimal. This is because there are 10.000 users in the upper neighbor whereas 50.000 users in the lower neighbor. Furthermore, there are 2.000 users in the left neighbor whereas 4.000 users in the right neighbor. Based on these neighbor cells' densities, it is likely that the bottom right corner of C_k is denser whereas the upper left corner is more sparse, because the intra-cell distribution is likely to be affected by neighbor cells. According to the original intuition behind adaptive grids [10,13], it is desirable to have many small cells in dense areas but few large cells in sparse areas. Hence, instead of dividing C_k evenly (as done in

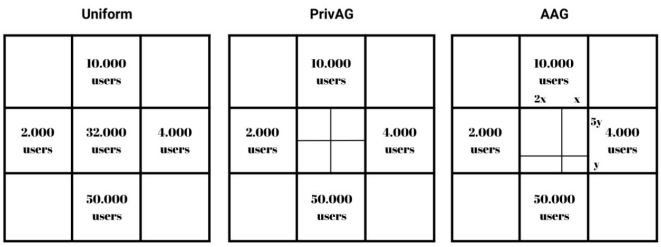


Fig. 1. Difference between PrivAG and AAG

PrivAG), AAG proposes to divide the cell by taking into account the neighbors' densities. Therefore, the vertical division of the cell is done with the ratio 1-to-5 which is proportional to the densities of the upper and lower neighbors (10.000 vs 50.000), whereas the horizontal division is done with the ratio 1-to-2 which is proportional to the left and right neighbors (2.000 vs 4.000). The result is shown on the right of Figure 1.

We give an algorithmic overview of our proposed AAG approach in Algorithm 3. The first phase of AAG is identical to PrivAG, i.e., it lays a $g_1 \times g_1$ uniform grid $\mathcal{G}_1$ on Ω and obtains the cell densities $\Phi(C_1)$, $\Phi(C_2)$, ... for all $C_k \in \mathcal{G}_1$ using ε-LDP. The core difference lies in the second phase. After computing g_2^k, instead of dividing cell C_k uniformly, AAG first calculates the horizontal split location using the densities of the left and right neighbors. Denoting the left neighbor of C_k by C_k^L and the right neighbor of C_k by C_k^R, the horizontal split location *hsplit* is calculated as:

$$hsplit = \frac{\Phi(C_k^R)}{\Phi(C_k^L) + \Phi(C_k^R)} \times (\text{width of } C_k) \tag{2}$$

Similarly, to calculate the vertical split location, the densities of the upper and lower neighbors are used. Denoting the upper neighbor of C_k by C_k^U and the lower neighbor of C_k by C_k^B, the vertical split location *vsplit* is calculated as:

$$vsplit = \frac{\Phi(C_k^B)}{\Phi(C_k^U) + \Phi(C_k^B)} \times (\text{height of } C_k) \tag{3}$$

Then, C_k is divided horizontally using *hsplit* and vertically using *vsplit*. As a result, four subcells of C_k are obtained. If $g_2^k > 2$, then each of the four subcells needs to be further divided. This further division is done uniformly, i.e., uniformly into $(g_2^k - 1)/2$ pieces horizontally and $(g_2^k - 1)/2$ pieces vertically, to make sure that C_k is divided into $g_2^k \times g_2^k$ subcells overall.

Handling edge and corner cells. When C_k is a cell that is located on one of the edges or corners of $\mathcal{G}_1$, it will lack one or more neighbors. For example, consider the top left cell in Figure 1, which is a corner cell. This cell has a lower neighbor and a right neighbor, therefore $\Phi(C_k^B)$ and $\Phi(C_k^R)$ can be found. However, it does not have an upper neighbor or a left neighbor, therefore $\Phi(C_k^U)$ and $\Phi(C_k^L)$ are not available. Similarly, if a cell is an edge cell, then it has three

Algorithm 3 Algorithmic summary of the AAG approach

Input: Users $\mathcal{U}$, parameters α and σ, privacy budget ε
Output: Densities of each cell in adaptive grid $\mathcal{G}_{aag}$

▷ **First phase of AAG**
The first phase of AAG is the same as PrivAG

▷ **Second phase of AAG**
for each cell $C_k \in \mathcal{G}_1$ **do**
    Compute *hsplit* and *vsplit* for C_k according to Equations 2 and 3
    Divide C_k into four subcells using *hsplit* and *vsplit*
    if $g_2^k > 2$ **then**
        Uniformly divide each subcell into $\frac{g_2^k - 1}{2}$ pieces horizontally and vertically
Let $\mathcal{G}_{aag}$ denote the resulting grid after the above divisions
Call Algorithm 1 with $\mathcal{U}_2$, $\mathcal{G}_{aag}$ and ε to obtain $\Phi(C_1)$, $\Phi(C_2)$, .. for all cells in $\mathcal{G}_{aag}$

neighbors but it lacks one neighbor. For cells that lack one or more neighbors, computing their *hsplit* and *vsplit* locations via Equations 2 and 3 using zero densities for the missing neighbors leads to erroneous results. To address this problem, we perform the following. If any of the neighbors of the current cell C_k is missing, then C_k uses its own density $\Phi(C_k)$ in place of the missing neighbor's density. For example, for the top left cell which lacks an upper neighbor and left neighbor, instead of assuming $\Phi(C_k^U) = 0$ and $\Phi(C_k^L) = 0$, we enforce: $\Phi(C_k^U) = \Phi(C_k)$ and $\Phi(C_k^L) = \Phi(C_k)$.

Choice of g_1 and g_2^k. PrivAG has two parameters (α and σ) which affect the values of g_1 and g_2^k. As we experimented with PrivAG and AAG, we observed that the recommended values for the α and σ parameters in PrivAG yield $\mathcal{G}_{aag}$ with cell counts that are similar to the initial uniform grid $\mathcal{G}_1$, which diminishes the benefits of using an adaptive grid. To address this problem, we propose to use different values for the α and σ parameters in AAG, leading to different choices of g_1 and g_2^k. Our choices aim to obtain an increased number of cell divisions in dense regions so that dense regions can be partitioned and represented in more detail, but without causing excessively large g_2^k. Specifically, as opposed to the default values of α and σ in PrivAG, we use $\alpha = 0.25$ and $\sigma = 0.5$ in AAG.

4 Experiments and Discussion

4.1 Experiment Setup and Datasets

In this section, we experimentally compare the three grid-based decomposition methods (UG, PrivAG, AAG). We implemented all algorithms and methods in Python. We use three real-world location datasets in our experiments: Gowalla, Porto, and Foursquare. To account for LDP randomness, each experiment is repeated 10 times and the average results are reported.

Gowalla: Gowalla was a location-based social networking site where users shared their locations via check-ins. From the full dataset, we extracted check-ins

made in the United States, between longitudes -124.26 and -71.87 and latitudes 25.45 and 47.44. Consequently, we have 3,451,190 remaining locations.

Porto: The Porto dataset contains trips of 442 taxis driving in the city of Porto. It was released as part of the Taxi Service Prediction Competition in ECML-PKDD. The original dataset contains full taxi trips, i.e., trajectories with multiple location readings per trip. We pre-processed it by keeping only one randomly sampled location from each trip and treated them as the current locations of users $\mathcal{U}$. We only used the locations between longitudes -8.691294 and -8.552009 and latitudes 41.138351 and 41.185935, corresponding to the city of Porto. This resulted in 1,620,157 remaining locations.

Foursquare: The Foursquare dataset contains location check-ins of social media users in Tokyo, between April 2012 and February 2013. We used this dataset without pre-processing. In total, the dataset contains 573,703 locations.

Following previous works, we use spatial density queries for utility measurement [1,8,10,13]. A spatial density query q with geospatial area denoted by $A(q)$ is a query of the form: "How many users are located within $A(q)$"? Let ans_q denote the ground truth answer of q and ans'_q denote the version estimated using LDP grids. We generate $\gamma = 500$ number of random queries $q_1, q_2, ...$ with different $A(q_i)$ and compute their ans_{q_i} and ans'_{q_i}. Then, we measure the average error between ans_{q_i} and ans'_{q_i} using the AQE metric:

$$AQE = \frac{1}{\gamma} * \sum_{i=1}^{\gamma} \frac{|ans_{q_i} - ans'_{q_i}|}{max\{ans_{q_i}, b\}} \tag{4}$$

Here, b denotes a bound to mitigate the dominating effect of queries with extremely low ans_{q_i} [8]. We set the value of b as: $b = 2\% \times |\mathcal{U}|$.

4.2 Comparison of Grid Approaches

In this section, we compare the three grid approaches (UG, PrivAG, AAG) using different-sized random queries. To do so, we enforce that the random queries we generate for calculating AQE have size: $A(q) = \rho \times \Omega$, where $\rho \in (0, 1]$ is the query size parameter. In Table 1, we fix $\varepsilon = 1$ and vary the value of ρ between 0.005% and 0.5%. Note that these are relatively low values of ρ, i.e., the generated queries are small. We use the **bold** notation in Table 1 when comparing the two adaptive grid approaches (PrivAG vs AAG), i.e., the one that yields lower AQE is written in bold. This helps to demonstrate the improvement of AAG compared to PrivAG. We use the grey cell background to denote the best-performing approach when comparing all three (UG vs PrivAG vs AAG).

The results in Table 1 show that AAG achieves lower error compared to PrivAG in all settings. In addition, AAG also achieves lower error compared to UG in the majority of settings. However, as ρ increases, UG starts performing better than AAG. This implies that for fine-grained density modeling and small-sized queries, AAG is the best approach. This is an intuitive result, considering that AAG excels in dividing dense areas in a detailed fashion. Yet, for more coarse (high-level) density statistics, UG can perform better. Another observation we

Table 1. AQEs with varying query sizes ρ, fixed $\varepsilon = 1$.

Dataset	Method	$\rho = 0.005\%$	$\rho = 0.01\%$	$\rho = 0.05\%$	$\rho = 0.1\%$	$\rho = 0.5\%$
Gowalla	UG	0.0034	0.0067	0.0279	0.0485	0.120
	PrivAG	0.0039	0.0077	0.0374	0.0728	0.305
	AAG	**0.0023**	**0.0051**	**0.0236**	**0.0460**	**0.185**
Porto	UG	0.0028	0.0045	0.0180	0.0321	0.082
	PrivAG	0.0034	0.0056	0.0283	0.0540	0.205
	AAG	**0.0025**	**0.0045**	**0.0247**	**0.0501**	**0.195**
Foursquare	UG	0.0032	0.0054	0.0243	0.0416	0.126
	PrivAG	0.0036	0.0062	0.0291	0.0547	0.203
	AAG	**0.0025**	**0.0043**	**0.0234**	**0.0450**	**0.177**

Table 2. AQEs with varying privacy budgets ε, fixed $\rho = 0.01\%$.

Dataset	Method	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 3$	$\varepsilon = 5$
Gowalla	UG	0.0070	0.0066	0.0064	0.0056
	PrivAG	0.0075	0.0077	0.0072	0.0062
	AAG	**0.0047**	**0.0051**	**0.0049**	**0.0041**
Porto	UG	0.0048	0.0045	0.0045	0.0045
	PrivAG	0.0058	0.0056	0.0059	0.0060
	AAG	**0.0048**	**0.0045**	**0.0049**	**0.0049**
Foursquare	UG	0.0055	0.0054	0.0051	0.0048
	PrivAG	0.0060	0.0062	0.0061	0.0069
	AAG	**0.0044**	**0.0043**	**0.0040**	**0.0047**

make from Table 1 is that when ρ increases, the errors also increase. This is because increasing ρ causes the intersection between $A(q_i)$ and various cells to increase, therefore ans_{q_i} and ans'_{q_i} become larger. Hence, overall noise amount increases as well, and among the two factors in the denominator of Equation 4, ans_{q_i} starts to dominate rather than b. Consequently, higher AQEs are obtained.

4.3 Impact of the Privacy Budget ε

In this section, we keep the query sizes ρ fixed and vary the privacy budgets ε between 0.5 and 5. We selected this range of ε values since they are parallel to the commonly used values in the LDP literature. Table 2 provides the results with $\rho = 0.01\%$ and Table 3 provides the results with $\rho = 4\%$. In both tables, we use the same bold and grey color highlight strategies that we used in the previous section. According to the results in Table 2, when $\rho = 0.01\%$, AAG is the best approach. It yields the lowest AQEs across all ε. On the other hand, when $\rho = 4\%$, UG becomes the best approach as shown in Table 3. This is parallel to the results reported in the previous section. When $\rho = 4\%$, although AAG consistently beats PrivAG, it cannot reach UG's low AQE values.

In both tables, we observe that as ε increases, AQEs of UG decrease. This is an intuitive result since higher ε means less perturbation caused by LDP, therefore results are more accurate. On the other hand, this trend does not always hold for PrivAG and AAG. For example, despite increasing ε, there are

Table 3. AQEs with varying privacy budgets ε, fixed $\rho = 4\%$.

Dataset	Method	$\varepsilon = 0.5$	$\varepsilon = 1$	$\varepsilon = 3$	$\varepsilon = 5$
Gowalla	UG	0.28	0.23	0.17	0.19
	PrivAG	1.09	1.15	1.08	0.98
	AAG	**0.68**	**0.71**	**0.68**	**0.76**
Porto	UG	0.16	0.12	0.10	0.09
	PrivAG	0.77	0.76	0.71	0.71
	AAG	**0.49**	**0.49**	**0.62**	**0.63**
Foursquare	UG	0.26	0.19	0.16	0.16
	PrivAG	0.69	0.63	0.75	0.69
	AAG	**0.61**	**0.56**	**0.55**	**0.51**

cases in PrivAG and AAG in which AQE values increase. This is because ε is used in the choice of g_1 and g_2^k. Hence, changing ε also changes the grid structures in PrivAG and AAG. These structural changes may affect ans'_q positively or negatively, and LDP perturbation is no longer the only factor in the accuracy of ans'_q. Thus, we do not see a consistent trend between ε and AQE in PrivAG and AAG. On the other hand, this observation shows that if the choices of g_1 and g_2^k are made in a more optimized fashion, especially in high ε regions, there is potential to improve utility, which can be an avenue for future work.

Combining all experiment results, we arrive at the following take-away messages: (i) AAG is preferable to PrivAG across all ε and ρ, (ii) AAG is the best approach when ρ is small, e.g., for computing answers to small queries or for detailed statistics, and (iii) UG is the best approach when ρ is large, e.g., for computing answers to large queries or for coarse statistics.

5 Conclusion

In this paper, we studied three grid-based decomposition approaches under LDP: UG, PrivAG, and AAG. Our proposed AAG approach advances the state-of-the-art adaptive grid approach (PrivAG) by performing cell divisions according to neighboring cells' densities. We experimentally compared UG, PrivAG, and AAG using three datasets and multiple ε and ρ values. We observed that AAG always beats PrivAG, and it also beats UG when ρ is small. However, when ρ is large, UG with a near-optimal choice of grid size becomes better than AAG. Note that the utility improvement of AAG comes at no additional LDP cost for users since UG, PrivAG, and AAG are compared using the same ε budgets.

Overall, considering the use of grids in the DP and LDP literature as well as the utility improvement offered by AAG, our work enables potential utility improvements in various LDP tasks such as density estimation and visualization, query answering, trajectory collection and sharing, and synthetic data generation [5,13,8]. In future work, we plan to integrate AAG into such downstream tasks. Furthermore, we plan to compare UG, PrivAG, and AAG against tree-based decompositions. Finally, we will investigate methods to improve PrivAG and AAG's utility especially in high ε regimes.

Acknowledgments

This study was supported by Scientific and Technological Research Council of Türkiye (TUBITAK) under Grant Number 121E303. The authors thank TUBITAK for their support.

References

1. Alptekin, E., Gursoy, M.E.: Building quadtrees for spatial data under local differential privacy. In: IFIP Annual Conference on Data and Applications Security and Privacy. pp. 22–39. Springer (2023)
2. Cormode, G., Jha, S., Kulkarni, T., Li, N., Srivastava, D., Wang, T.: Privacy at scale: Local differential privacy in practice. In: Proceedings of the 2018 International Conference on Management of Data. pp. 1655–1658. ACM (2018)
3. Cormode, G., Maddock, S., Maple, C.: Frequency estimation under local differential privacy. Proceedings of the VLDB Endowment **14**(11), 2046–2058 (2021)
4. Ding, B., Kulkarni, J., Yekhanin, S.: Collecting telemetry data privately. In: Advances in Neural Information Processing Systems. pp. 3571–3580 (2017)
5. Du, Y., Hu, Y., Zhang, Z., Fang, Z., Chen, L., Zheng, B., Gao, Y.: Ldptrace: Locally differentially private trajectory synthesis. Proceedings of the VLDB Endowment **16**(8), 1897–1909 (2023)
6. Erlingsson, Ú., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. pp. 1054–1067. ACM (2014)
7. Gursoy, M.E., Liu, L., Chow, K.H., Truex, S., Wei, W.: An adversarial approach to protocol analysis and selection in local differential privacy. IEEE Transactions on Information Forensics and Security **17**, 1785–1799 (2022)
8. Gursoy, M.E., Liu, L., Truex, S., Yu, L., Wei, W.: Utility-aware synthesis of differentially private and attack-resilient location traces. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 196–211 (2018)
9. Hong, D., Jung, W., Shim, K.: Collecting geospatial data under local differential privacy with improving frequency estimation. IEEE Transactions on Knowledge and Data Engineering (2022)
10. Qardaji, W., Yang, W., Li, N.: Differentially private grids for geospatial data. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE). pp. 757–768. IEEE (2013)
11. Wang, H., Hong, H., Xiong, L., Qin, Z., Hong, Y.: L-srr: Local differential privacy for location-based services with staircase randomized response. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 2809–2823. Association for Computing Machinery, New York, NY, USA (2022)
12. Wang, T., Blocki, J., Li, N., Jha, S.: Locally differentially private protocols for frequency estimation. In: Proc. of the 26th USENIX Security Symposium. pp. 729–745 (2017)
13. Yang, J., Cheng, X., Su, S., Sun, H., Chen, C.: Collecting individual trajectories under local differential privacy. In: 2022 23rd IEEE International Conference on Mobile Data Management (MDM). pp. 99–108. IEEE (2022)
14. Zhang, Y., Ye, Q., Chen, R., Hu, H., Han, Q.: Trajectory data collection with local differential privacy. Proceedings of the VLDB Endowment **16**(10), 2591–2604 (2023)

Balancing Privacy and Utility in Multivariate Time-Series Classification

Adrian-Silviu Roman[1 [0000−0002−2246−3415]], Béla Genge[1 [0000−0003−1390−479X]], and Piroska Haller[1 [0000−0002−5611−2429]]

George Emil Palade University of Medicine, Pharmacy, Science and Technology of Targu Mures, Targu Mures, Mures, 540142, Romania
{adrian.roman, bela.genge, piroska.haller}@umfst.ro

Abstract. In the modern era, characterized by the widespread presence of sensor-based systems, ensuring time-series data privacy without compromising utility has emerged as a significant challenge. While current strategies for safeguarding time-series data prioritize either input data protection or privacy-preserving classification models, they often fall short in assessing the balance between data privacy and utility, particularly when strong adversary classification models are involved. Our approach introduces a novel protection technique specifically designed for multivariate Time-Series Classification. This technique involves perturbing the data by distributing the noise among features using a feature importance-based approach, thus securing the data while preserving its analytical value. We propose a dual-model evaluation system consisting of two supervised classifiers, a Privacy-Breaking Classifier and a Utility-Focused Classifier. These are designed to respectively assess the potential for privacy breaches and the extent of data utility preservation in the context of protected time-series data. The experimental results demonstrate the effectiveness and viability of our methodology. Our approach provides a framework for evaluating the privacy and utility levels of time-series data. Additionally, it guides the selection of an appropriate perturbation level to ensure both aspects are adequately addressed.

Keywords: Utility-preserving data privacy · Time-series classification · Local differential privacy · Automotive systems.

1 Introduction

Data collected from sensor-based systems, such as automotive vehicles, wearable devices, or smart grids, is often transmitted over the Internet to centralized databases for analysis and processing by third-party systems (e.g., traffic monitoring conducted by authorities or insurance companies). While time-series data itself does not explicitly contain personally identifiable information (e.g., names, e-mails), they may expose user-related details (e.g., geolocation and biometrics) or lead to user identification or re-identification (e.g., by data classification) [6, 12]. Protecting sensor data from both "honest-but-curious" data processors and malicious actors is crucial. To address these concerns, data can be distorted or aggregated to enforce privacy without losing its utility.

While methods exist to safeguard data collected from sensors during transmission to third-party systems [10], ensuring data privacy poses additional challenges. When applying data protection mechanisms, monitoring and retaining data usefulness is essential to ensure that enough information is preserved for effective analysis. Typically, evaluating data utility (i.e., the usefulness of data) involves measuring how well a selected data privacy method preserves aggregate statistical information. However, when addressing the classification of time-series data, relying exclusively on aggregate statistical information derived from protected data might not adequately capture the extent to which the data achieves a balance between privacy and utility. In the field of time-series data classification, data privacy predominantly relies on two primary methodologies: (i) secure input data through diverse techniques such as perturbation, encryption, de-identification, data transformation, machine learning methodologies (e.g., using deep autoencoders) [11, 14]; and (ii) employing classification models that preserve privacy, thereby securing the training data against potential adversaries [1].

The proposed approach introduces a time-series data protection mechanism tailored for multivariate data, which can be implemented at the device level and is designed to neutralize adversaries using powerful classifiers. This mechanism is configurable to achieve the desired balance between privacy and utility. To summarize, the research presented in this paper advances the state of the art from several perspectives:

- We formulate the problem of balancing privacy and utility in the context of multivariate Time-Series Classification (TSC), using a dual-model, consisting of two opposing classifiers, a Privacy-Breaking Classifier (PBC) and a Utility-Focused Classifier (UFC);
- We propose a protection technique independent of the perturbation type, applying the perturbation to multivariate time-series data and utilizing feature importance to distribute the noise;
- We introduce the *classification utility-privacy balance* score, $\mathcal{B}_{UP}$, and a methodology for determining the appropriate value of the applied perturbation, to achieve the desired level of utility and privacy;
- We demonstrate the effectiveness and viability of the proposed methodology by enforcing a Local Differential Privacy (LDP) perturbation on two well-known driver datasets [9, 15], and compare the results with the outcomes of uniformly applying the same perturbation across all features.

The remainder of this paper is organized as follows. Section 2 describes the addressed problem, and it introduces the basic concepts and terminology. Section 3 presents an in-depth description of the proposed approach. This is followed by extensive experimental results in Section 4. The paper concludes in Section 5.

2 Problem Definition and Basic Concepts

Consider data collected from sensors, temporarily stored at the device level, and sent as data streams to a central data warehouse for classification. The objective

is to protect the multivariate time-series data from a privacy-breaking adversary classifier while preserving data utility. Furthermore, we impose the restriction that no dimensionality reduction methods are permitted, ensuring that the full set of features is maintained.

Within this setting, two distinct types of classifiers are evaluated: (i) a UFC, for identifying non-user related data patterns, and (ii) a PBC, designed to determine the identity of the users. The objective of this research is to propose a protection mechanism at the device level, such that the UFC provides suitable detection accuracy while the accuracy of the PBC is as low as possible.

Classification model generation The process of generating the UFC model assumes a secure environment without any device-level perturbations. In both the training and testing phases of constructing the UFCs, data labels for each batch of records are stored in the data warehouse. After constructing the UFC models, there is the possibility to remove the data labels from the database. No label information is shared with third-party systems. Moreover, during the prediction phase, no explicit information that could identify the user is transmitted to the central database, or recorded in any form. This procedure serves as a fundamental privacy protection strategy.

Adversarial model We examine the scenario in which the attacker possesses access to all potential data sources, containing both the training dataset with labels, the unlabeled data collected from the devices, and other observable, non-perturbed data. Consequently, the attacker is capable of constructing a PBC that promptly and accurately re-identifies users based on intercepted data. Another potential privacy breach is associated with "honest-but-curious" entities, such as a data analyst who has access to all the data needed to construct a classifier that compromises privacy.

Local w-event level differential privacy for time series To demonstrate the proposed approach, we choose LDP as the time series perturbation method. This involves applying LDP perturbation per feature for each batch of w data values. The *w-event-level* DP perturbation [8] provides privacy protection to any sequence of w consecutive events and represents a compromise between the *user* and the *event-level privacy*. To perturb the data at the device level we utilize the local *w-event-level* DP [13]. The size of the noise is determined by the privacy budget ϵ and the sensitivity [3]. Introducing noise into the data through a mechanism that satisfies ϵ-differential privacy is achieved using the Laplace mechanism, as proposed by Dwork et al. [3]. Data collected from sensors is multivariate, meaning the resulting time-series data has numerous attributes or features. We apply the w-event LDP separately for each feature.

Privacy and utility measurements To validate the proposed approach in the context of TSC, we use the classification accuracy of the PBC as a measure of privacy protection. Similarly, the classification accuracy of the UFC measures the data utility. Additionally, we employ the Mean Average Error (MAE) to further assess the utility of the perturbed data, a metric commonly used for

comparing time series perturbation methods [16]: $MAE = \frac{1}{N} \quad _{i \in N} |V_i - V_i'|$, where N is the number of values of the time series, V is the original set of values, and V' is the perturbed data.

3 Proposed Approach

Time-series data, continuously collected from sensors, is perceived as infinite data streams, with each distinct value representing an event generated by the user (such as an event indicating "the driver u traveled at speed s at time t"). Further, we introduce our proposed approach for protecting the sensitive data encapsulated in these events.

We propose a time-series perturbation method suitable for multivariate time-series data that outperforms the uniform distribution of perturbation across all features. Our approach is designed to be independent of the specific perturbation method used, ensuring robustness and flexibility in handling diverse types of time-series data. Additionally, the method is developed for locally applied perturbation at the device level.

Let the continually collected data, consisting of records with d features, be stored locally on the device and then transmitted in groups of w records to a central location for processing. Thus, let $\mathbf{X}_i^t$ be the time-series data collected for a period of time and stored locally at time t, on the sensor-based device i. $\mathbf{X}_i^t$ is represented as a matrix of w rows and d columns. Each column denotes measurements associated with a particular attribute/feature.

TSC refers to the task of assigning a label or category to a given time-series data based on its patterns, trends, or features. A classifier model produces a function $f : \mathbb{R}^{w \cdot d} \longrightarrow \mathcal{C}, \mathcal{C} = \{1, ..., n\}$, where w is the number of records, d is the number of features and n the number of categories, that is applied to any input $\mathbf{X}$. The function $f(\mathbf{X})$ is regarded as an estimate of the category that $\mathbf{X}$ belongs to [5]. Let f_p and f_u be two classification functions, $f_p : \mathbb{R}^{w \cdot d} \longrightarrow \mathcal{C}_u$ and $f_u : \mathbb{R}^{w \cdot d} \longrightarrow \mathcal{C}_p$ that output the class that $\mathbf{X}_i^t$ belongs to, in case of the PBC, and UFC, respectively. $f_p(\mathbf{X}_i^t)$ is the PBC classifier that breaks the user privacy by identifying the set of collected values as belonging to a user from a specified class $\mathcal{C}_p$, and $f_u(\mathbf{X}_i^t)$, the UFC, classifies the non-sensitive information from a class $\mathcal{C}_u$.

In the training and testing phases of building the UFC, for each $\mathbf{X}_i^t$ an id of the device or/and of the user is also collected and stored, with the objective of labeling the data. Let $\mathbf{Y}_{pi}^t$ be the label information for the user identification, and let $\mathbf{Y}_{ui}^t$ be the label for non-user-related data classification. In the prediction phases, as a fundamental measure to preserve privacy, no data that could identify individuals or devices is transmitted to the central data warehouse.

The objective is to perturb time-series data $\mathbf{X}_i^t$ with a mechanism $\mathcal{M}$ such that two conditions are met: the classification accuracy of f_p on the perturbed data is minimum, and, simultaneously, the classification accuracy of f_u on the perturbed data is maximum.

Let the classification accuracy, which shows the number of correct classifications, for the two classifiers f_u and f_p be defined as follows:

$$\mathcal{A}_u = \frac{1}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{I}(\mathbf{Y}_{ui}^t = f_u(\mathbf{X}_i^t)), \mathcal{A}_p = \frac{1}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{I}(\mathbf{Y}_{pi}^t = f_p(\mathbf{X}_i^t)), \quad (1)$$

where $\mathcal{A}_u$ is the accuracy of f_u, $\mathcal{A}_p$ is the accuracy of f_p, N is the number of sensor-based devices, T the number of data batches collected, and $\mathbb{I}(c)$ the binary indicator function returning 1 iff the condition c is true, and returns 0 otherwise. Furthermore, the classification accuracy for the perturbed data is defined as:

$$\mathcal{A}'_u(\theta) = \frac{1}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{I}(\mathbf{Y}_{ui}^t = f_u(\mathcal{M}(\mathbf{X}_i^t; \theta))), \text{and} \quad (2)$$

$$\mathcal{A}'_p(\theta) = \frac{1}{N \cdot T} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{I}(\mathbf{Y}_{pi}^t = f_p(\mathcal{M}(\mathbf{X}_i^t; \theta))). \quad (3)$$

where $\mathcal{M}$ is the perturbation mechanism, $\mathcal{M} : \mathbb{R}^{w \cdot d} \longrightarrow \mathbb{R}^{w \cdot d}$, and θ is the set of perturbation parameters (e.g., ϵ for LDP).

Thus, the data perturbation objective is bounded by the following conditions:

$$\mathcal{A}'_p(\theta) \ll \mathcal{A}_p \text{ and } \mathcal{A}'_u(\theta) \approx \mathcal{A}_u, \quad (4)$$

such that the accuracy of PBC is significantly diminished when data is perturbed, while the accuracy of the UFC for perturbed data remains close to the accuracy of the classification on non-perturbed data.

Achieving a balance between privacy and utility presents a significant challenge, particularly when considering classification tasks involving two opposing classifiers. We introduce $\mathcal{B}_{UP}$, the *classification utility-privacy balance*, as a measure for balancing privacy and utility in the proposed classification problem, computed as follows:

$$\mathcal{B}_{UP}(\theta) = 1 - \frac{\mathcal{A}'_u(\theta)}{\mathcal{A}_u} \cdot \left(1 - \frac{\mathcal{A}'_p(\theta)}{\mathcal{A}_p}\right). \quad (5)$$

The score $\mathcal{B}_{UP}$ is positive and close to zero when the perturbation successfully meets the specified objectives from Eq. 4. This indicates that the accuracy of the UFC with perturbed data is close to its accuracy with unperturbed data, while the accuracy of the PBC with perturbed data is significantly lower than its accuracy with unperturbed data. A negative $\mathcal{B}_{UP}$ value signifies that the chosen perturbation has increased the accuracy of the PBC, representing the worst-case scenario.

The proposed method for finding the perturbation parameters consists of the following steps: (i) compute feature importance for the two classifications (UFC and PBC); (ii) cluster features based on the computed importance coefficients; (iii) distribute and apply the perturbation to the features of the time-series data $\mathbf{X}_i^t$; (iv) select the perturbation parameter set θ^* such that $\mathcal{B}_{UP}(\theta)$ is minimum:

$$\theta^* = argmin_\theta \{\mathcal{B}_{UP}(\theta) | \mathcal{B}_{UP}(\theta) > 0\}. \quad (6)$$

3.1 Feature Importance Computation and Feature Clustering

Feature importance computation [2] estimates the relative significance of features within a dataset for a specific classification. One notable benefit of employing feature importance computation techniques lies in assessing feature significance without requiring prior knowledge of the adversarial model. Consequently, this computation is conducted solely based on available data labels and by anticipating potential classifications.

We propose using the Random Forest (RF) algorithm [2], a popular technique for feature ranking and feature importance computation, to identify the features that hold the most relevance for classifications. For the current research, we selected the impurity-based feature importance (e.g., Gini importance [2]).

Let $\mathbf{X}$ be the set of all $\mathbf{X}_i^t$ selected for the training phase. By computing the feature importance for classifying $\mathbf{X}$ on class $\mathcal{C}_p$ using the labels $\mathbf{Y}_p$ and for classifying $\mathbf{X}$ on class $\mathcal{C}_u$ using labels $\mathbf{Y}_u$, we obtain I_p and I_u, the vectors of importance coefficients ($I_{pi}, I_{ui} \in [0,1], i \in \{1,...,d\}$).

Let $\mathcal{F}$ be the set of selected features from the dataset, $\mathcal{F} = \{F_1, ..., F_i, ...\}$. Based on the computed importance coefficients I_p and I_u, we aim to cluster $\mathcal{F}$ into three subsets: $\mathcal{F}_u$ with features necessary for the utility-focused classification, $\mathcal{F}_p$ with features that play a significant role in resolving the privacy-breaking classification, and $\mathcal{F}_{up}$ with features that are important for both classifications, such that $\mathcal{F} = \mathcal{F}_u \cup \mathcal{F}_p \cup \mathcal{F}_{up}$.

Let $S_{F_i}(I_{ui}, I_{pi})$ be the score of importance coefficients for feature F_i in I_p and I_u, respectively, such that: $S_{F_i}(I_{ui}, I_{pi}) = I_{ui} - I_{pi}$. Further, let the line represented by $S_{F_i}(I_{ui}, I_{pi}) = 0$, $\forall i \in \{1,...,d\}$, be the main decision boundary and ρ_I be the distance between the main and secondary decision boundaries. The following rules for clustering are considered:

$$F_i \in \mathcal{F}_u \text{ if } S_{F_i}(I_{ui}, I_{pi}) > \rho_I, F_i \in \mathcal{F}_p \text{ if } S_{F_i}(I_{ui}, I_{pi}) < -\rho_I, \tag{7}$$

$$F_i \in \mathcal{F}_{up} \text{ if } S_{F_i}(I_{ui}, I_{pi}) \in [-\rho_I, \rho_I]. \tag{8}$$

The score $S_{F_i}(I_{ui}, I_{pi})$ determines when a feature shows a significantly greater significance in a classification context compared to another. When $S_{F_i}(I_{ui}, I_{pi}) = 0$, only the features that have the same importance coefficients for both UFC and PBC belong to $\mathcal{F}_{up}$.

3.2 Data Perturbation

The proposed mechanism introduces a global perturbation budget to be divided within the features in $\mathcal{F}$. To achieve the goal of reducing the PBC's impact while preserving the accuracy of the UFC, we introduce substantial perturbation to features within $\mathcal{F}_p$, moderate perturbation to features within $\mathcal{F}_{up}$, while maintaining the features within the cluster $\mathcal{F}_u$ unmodified.

Consider the mechanism $\mathcal{M}$ of perturbing $\mathbf{X}_i^t$ a composition of d mechanisms $\mathcal{M}_i$, one for each feature F_i in clusters $\mathcal{F}_p$, $\mathcal{F}_{up}$ and $\mathcal{F}_u$. Let the total perturbation budget β_T be allocated to $\mathcal{M}$ and distributed to mechanisms $\mathcal{M}_i$, based on the

membership in clusters $\mathcal{F}_p$, $\mathcal{F}_{up}$ and $\mathcal{F}_u$. Let β_p, β_{up}, and β_u be the cumulative privacy budgets for features in clusters $\mathcal{F}_p$, $\mathcal{F}_{up}$ and $\mathcal{F}_u$, respectively, such that:

$$\beta_p = \alpha_p \cdot \beta_T, \beta_{up} = \alpha_{up} \cdot \beta_T, \beta_u = 0, \text{and } \alpha_p + \alpha_{up} = 1, \text{ with } \alpha_p, \alpha_{up} \in [0, 1], \quad (9)$$

where α_p and α_{up} are the perturbation budget distribution parameters. Within each cluster, budgets are uniformly distributed such that: $\beta_{pi} = \frac{\alpha_p \cdot \beta_T}{|\mathcal{F}_p|}$, and $\beta_{upi} = \frac{\alpha_{up} \cdot \beta_T}{|\mathcal{F}_{up}|}$, where β_{pi} is the privacy budget allocated for each $F_i \in \mathcal{F}_p$, β_{upi} the budget allocated for each $F_i \in \mathcal{F}_{up}$, and $|\mathcal{F}_p|$, $|\mathcal{F}_{up}|$ the number of features in clusters $\mathcal{F}_p$ and $\mathcal{F}_{up}$. Additionally, the imposed requirement is that the perturbation applied to features in $\mathcal{F}_p$ be higher or equal to the one applied to features in $\mathcal{F}_{up}$. When the data perturbation is proportional to the allocated budget we enforce that $\beta_{pi} \geq \beta_{upi}$, otherwise $\beta_{pi} \leq \beta_{upi}$.

A possible perturbation method to be applied in the proposed context involves implementing the w-event level LDP (described in Section 2) to features in clusters $\mathcal{F}_p$ and $\mathcal{F}_{up}$. The selection of perturbation budget parameters α_p and α_{up} needs to account for the number of features in the cluster. The challenge lies in identifying ρ_I, which defines the feature clustering (Section 3.1), along with β_T, α_p, α_{up}, to ensure that $\mathcal{B}_{UP}(\theta)$, with $\theta = \{\rho_I, \beta_T, \alpha_p, \alpha_{up}\}$, is minimized while remaining positive. Further, the experiments confirm that suitable values for the perturbation parameters can be identified and that the $\mathcal{B}_{UP}$ score effectively indicates the appropriate size of the perturbation.

4 Experimental Results

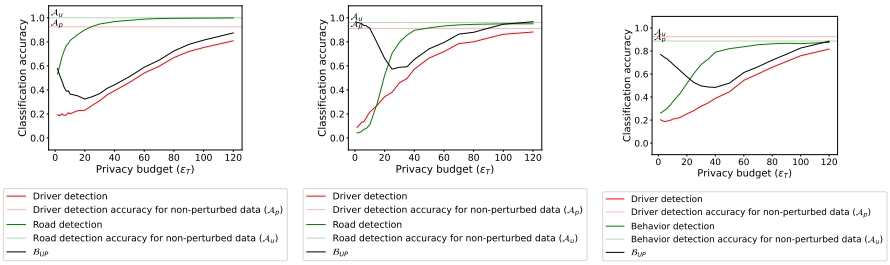
To demonstrate the validity of the proposed approach, we focused the experiments on two datasets collected from automotive vehicles, the UAH-Driveset [15], and, the HCRL Driving Dataset [9]. For benchmarking the driver identification classification, we utilized the LSTM-based approach, as proposed by Karim *et al.* [7].

Recall that the work at hand addresses two objectives: to protect sensitive information while concurrently preserving data utility in the context of multivariate TSC. We examine two distinct scenarios: (i) driver detection as PBC with road type detection as UFC, which we term *driver-vs-road-type classification*; and (ii) driver detection functioning as the PBC and behavior detection operating as the UFC, a setup we refer to as *driver-vs-behavior classification*.

The first experimental stage involved constructing classification models specifically trained to identify the driver, determine driver behavior, and categorize road types. These models utilized the FCN-LSTM architecture [7] and were trained on non-perturbed data from the selected datasets. The datasets were preprocessed following the same procedures proposed by El Mekki *et al.* [4]. The obtained results represented the accuracy achieved by the classification models in various tasks.

To evaluate the impact of distributing perturbation unequally between features, as outlined in Section 3.2, we initially explored a scenario where features

were not clustered, with each feature receiving an identical amount of perturbation. This approach was a direct extension of perturbation methods originally designed for univariate data, specifically the w-event LDP [8, 13]. In this case, $\epsilon_i = \epsilon_j$ for all $i, j \in \{1, ..., d\}$, and the total privacy budget for all features was $\epsilon_T = \sum_{i=0}^{d} \epsilon_i$. The budget ϵ_i for each feature F_i was uniformly distributed among the values belonging to the same feature, according to the formula $w \cdot \Delta g/\epsilon_i$ proposed by Kellaris et al. [8] for w-event LDP, where Δg represents the data sensitivity. The noise was generated using the Laplace mechanism [3] and applied to data batches of $w = 60$ records of sensor values. Figure 1 and Table 1 demonstrate that applying an equal perturbation to all features fulfills the objectives stated in Eq. 4. However, in all cases, the accuracy $\mathcal{A}'_p(\theta)$ of the PBC remained close to $\mathcal{A}'_u(\theta)$ for smaller perturbations, where $\mathcal{A}'_u(\theta) \approx \mathcal{A}_u$.



(a) Driver-vs-road-type (UAH Driveset)

(b) Driver-vs-road-type (HCRL Dataset)

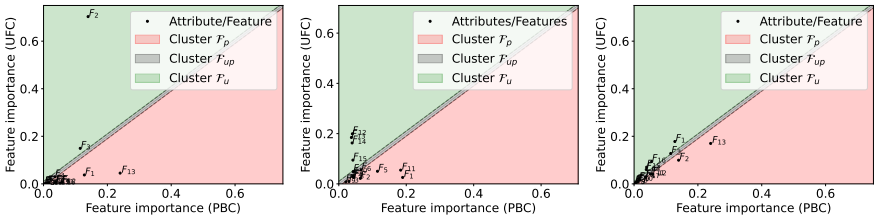
(c) Driver-vs-behavior (UAH Driveset)

Fig. 1: Classification accuracy with equally distributed perturbation for all features.

Furthermore, the experiments involved computing feature importance to evaluate the relevance of each feature for driver detection and non-driver-related data classification. Subsequently, features were clustered (Fig. 2) in $\mathcal{F}_p$, $\mathcal{F}_{up}$, and $\mathcal{F}_u$, according to the rules described in Section 3.1, considering the computed feature importance for driver identification and behavior/road type detection, respectively. Features in $\mathcal{F}_p$ and $\mathcal{F}_{up}$ were perturbed, as described in Section 3.2, using the same w-event LDP method, with variable privacy budget for each cluster.

The proposed approach for distributing the perturbation is based on a set of parameters, including the clustering parameter ρ_I, the privacy budget ϵ_T, and the cluster privacy budget distribution parameters α_p and α_{up}. We analyzed the number of features in each cluster to determine potential values for ρ_I. Our criteria ensured that there is at least one feature in both $\mathcal{F}_u$ and $\mathcal{F}_p$ and that the number of features in $\mathcal{F}_{up}$ is less than the sum of the features in $\mathcal{F}_p$ and $\mathcal{F}_u$ to maintain manageable uncertainty. For the cluster privacy budget distribution parameters α_p and α_{up}, we followed the condition that $\frac{\alpha_p}{|\mathcal{F}_p|} \leq \frac{\alpha_{up}}{|\mathcal{F}_{up}|}$. The considered values for ρ_I, α_p, and α_{up} are listed in Table 1.

The perturbed data was verified against previously trained classification models to determine an appropriate perturbation level ϵ_T, aiming to achieve the de-



(a) Driver-vs-road-type (UAH Driveset) (b) Driver-vs-road-type (HCRL Dataset) (c) Driver-vs-behavior (UAH Driveset)

Fig. 2: Feature clustering based on feature importance coefficients for two classifications (UFC and PBC), conducted using Random Forest with Gini importance ($\rho_I = 0.01$).

sired levels of privacy and utility while adhering to the established constraints. Figure 3 shows the classification accuracy for the considered parameter values in the case of *driver-vs-road-type classification*. The findings, as presented in Table 1 (with the best results highlighted in bold), demonstrated that certain parameter settings for the proposed approach produced superior outcomes compared to uniformly distributing noise across all features, aligning with the objectives outlined in Eq. 4. The overall performance evaluation was based on the calculation of the classification utility-privacy balance, $\mathcal{B}_{UP}(\theta)$.

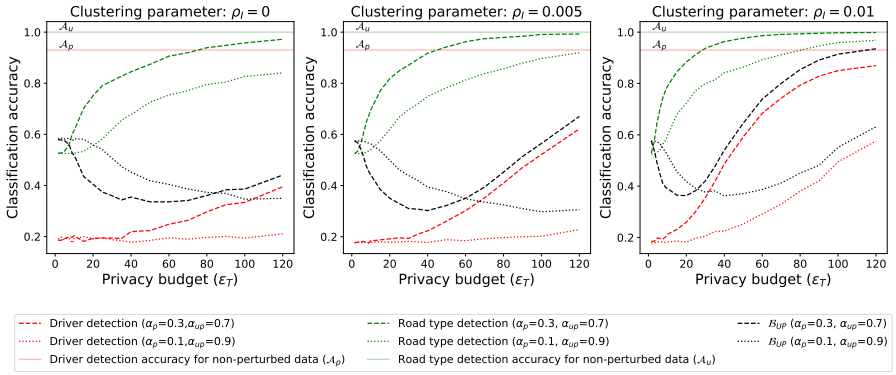


Fig. 3: Driver-vs-road-type classification accuracy on perturbed data (UAH dataset).

For the *driver-vs-road-type* classification using the UAH dataset, the accuracy $\mathcal{A}'_p(\theta)$ of the PBC was lower than the reference accuracy, while the classification accuracy $\mathcal{A}'_u(\theta)$ of the UFC is higher. The minimum value of $\mathcal{B}_{UP}(\theta)$ was achieved when clustering resulted in a large number of features in $\mathcal{F}_p$ and a small number of features in $\mathcal{F}_{up}$ and $\mathcal{F}_u$. This configuration allocated most of the perturbation to the features in $\mathcal{F}_p$, amplified by the large number of features in this cluster. Additionally, the MAE value was lower than the reference, indicating higher data utility. The privacy budget ϵ_T in this scenario was significantly higher ($\epsilon_T = 100$

compared to $\epsilon_T = 15$) than the proposed approach, indicating an overall lower perturbation. For the HCRL dataset, in the same type of classification, the best results were obtained when the clustering ($\rho_I = 0$) did not include features in $\mathcal{F}_{up}$.

In the case of the *driver-vs-behavior* classification, both classifiers relied on a similar set of features (Fig. 2c). However, the results showed that data protection was possible under the considered constraints with $\rho_I = 0$, $\alpha_p = 0.3$, $\alpha_p = 0.7$, and $\epsilon_T = 50$. Similar accuracy for PBC and UFC were obtained for significantly lower perturbation ($\epsilon_T = 50$ compared to $\epsilon_T = 30$).

5 Conclusion

In conclusion, our research aimed to reach a balance between data privacy and utility when dealing with protected time-series data in the context of TSC. The proposed methodology offers several significant advantages: it results in a lower $\mathcal{A}'_p$ (accuracy of PBC), a higher $\mathcal{A}'_u$ (accuracy of UFC), and improved utility, as quantified by MAE. Further benefits of this approach are: (i) certain features ($\mathcal{F}_u$, crucial for utility classification) remain unperturbed, thereby preserving their utility for a variety of data processing activities beyond classification; (ii) the perturbation budget is significantly higher, resulting in lower noise when clustering is employed to achieve comparable privacy and utility objectives.

Table 1: Classification accuracy for perturbed data using the proposed approach.

Dataset	**Classification scenario**	**Perturbation approach**	**Clustering parameter** (ρ_I)	**# of features per cluster** ($\mathcal{F}_p$, $\mathcal{F}_{up}$, $\mathcal{F}_u$)	**Perturbation parameters** (α_p, α_{up})	**min**($\mathcal{B}_{UP}$)	ϵ_T	$\mathcal{A}'_p(\theta)$	$\mathcal{A}'_u(\theta)$	**MAE**
UAH [15]	Driver-vs-road-type	w-event LDP [8, 13] (no clustering)	-	-	-	0.3348	15	0.2103	0.8610	1.0170
		w-event LDP (feature clustering, **proposed method**)	0	{15,0,2}	{0.3,0.7}	0.3357	60	0.2480	0.9058	0.7424
					{0.1,0.9}	0.3459	100	0.1941	0.8266	1.3350
			0.005	{12,3,2}	{0.3,0.7}	0.3022	40	0.2227	0.9175	0.6038
					{0.1,0.9}	**0.2973**	**100**	0.2012	0.8967	0.6921
			0.01	{10,5,2}	{0.3,0.7}	0.3630	20	0.2564	0.8831	0.8141
					{0.1,0.9}	0.3623	40	0.2253	0.8415	1.0194
HCRL [9]	Driver-vs-road-type	w-event LDP [8, 13] (no clustering)	-	-	-	0.5683	30	0.4356	0.7941	0.5291
		w-event LDP (feature clustering, **proposed method**)	0	{10,0,5}	{0.3,0.7}	0.5765	60	0.4613	0.8235	0.4306
					{0.15,0.85}	**0.5629**	**120**	0.4377	0.8076	0.4302
			0.01	{9,2,4}	{0.3,0.7}	0.5785	60	0.4635	0.8235	0.4304
					{0.15,0.85}	0.5749	100	0.3755	0.6945	0.5168
			0.015	{7,4,4}	{0.3,0.7}	0.5803	40	0.3841	0.6968	0.4946
					{0.15,0.85}	0.5707	90	0.4263	0.7750	0.4428
UAH [15]	Driver-vs-behavior	w-event LDP [8, 13] (no clustering)	-	-	-	0.4859	30	0.3090	0.6844	0.5077
		w-event LDP (feature clustering, **proposed method**)	0	{9,0,8}	**{0.3,0.7}**	**0.4692**	**50**	0.3181	0.7220	0.3651
					{0.15,0.85}	0.4834	100	0.3305	0.7175	0.3655
			0.005	{6,6,5}	{0.3,0.7}	0.4986	20	0.3759	0.7545	0.4110
					{0.15,0.85}	0.5426	20	0.4000	0.7201	0.5002
			0.01	{4,8,5}	{0.3,0.7}	0.4836	25	0.3714	0.7707	0.3609
					{0.15,0.85}	0.4965	20	0.3480	0.7207	0.4458

We introduced the *classification utility-privacy balance* score, $\mathcal{B}_{UP}$, which provides a detailed assessment of the privacy and utility achieved by the selected

parameter set. This score maximizes the distance between $\mathcal{A}'_u(\theta)$ and $\mathcal{A}'_p(\theta)$, effectively balancing the trade-offs. However, alternative perturbation parameters may be selected if the accuracy of PBC or UFC is prioritized. Depending on specific requirements, such as maximizing the classification accuracy for PBC or maintaining the classification accuracy for UFC below a predetermined threshold, different perturbation parameters can be chosen.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. pp. 308–318 (2016)
2. Breiman, L.: Random forests. Machine Learning **45**, 5–32 (2001)
3. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Theory of cryptography conference. pp. 265–284. Springer (2006)
4. El Mekki, A., Bouhoute, A., Berrada, I.: Improving driver identification for the next-generation of in-vehicle software systems. IEEE Transactions on Vehicular Technology **68**(8), 7406–7415 (2019)
5. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
6. Hallac, D., Sharang, A., Stahlmann, R., Lamprecht, A., Huber, M., Roehder, M., Sosic, R., Leskovec, J.: Driver identification using automobile sensor data from a single turn. pp. 953–958 (11 2016)
7. Karim, F., Majumdar, S., Darabi, H., Chen, S.: Lstm fully convolutional networks for time series classification. IEEE access **6**, 1662–1669 (2017)
8. Kellaris, G., Papadopoulos, S., Xiao, X., Papadias, D.: Differentially private event sequences over infinite streams. Proc. VLDB Endow. **7**(12), 1155–1166 (aug 2014)
9. Kwak, B.I., Woo, J., Kim, H.K.: Know your master: Driver profiling-based anti-theft method. In: PST 2016 (2016)
10. Lyu, L., He, X., Law, Y.W., Palaniswami, M.: Privacy-preserving collaborative deep learning with application to human activity recognition. pp. 1219–1228 (11 2017)
11. Malekzadeh, M., Clegg, R.G., Cavallaro, A., Haddadi, H.: Mobile sensor data anonymization. In: Proceedings of the international conference on internet of things design and implementation. pp. 49–58 (2019)
12. Mekruksavanich, S., Jitpattanakul, A.: Biometric user identification based on human activity recognition using wearable sensors: An experiment using deep learning models. Electronics **10**(3), 308 (2021)
13. Ren, X., Shi, L., Yu, W., Yang, S., Zhao, C., Xu, Z.: Ldp-ids: Local differential privacy for infinite data streams. In: Proceedings of the 2022 international conference on management of data. pp. 1064–1077 (2022)
14. Roman, A.S., Genge, B., Duka, A.V., Haller, P.: Privacy-preserving tampering detection in automotive systems. Electronics **10**(24) (2021)
15. Romera, E., Bergasa, L.M., Arroyo, R.: Need data for driver behaviour analysis? presenting the public uah-driveset. In: 2016 IEEE 19th international conference on intelligent transportation systems (ITSC). pp. 387–392. IEEE (2016)
16. Wang, H., Xu, Z.: Cts-dp: publishing correlated time-series data via differential privacy. Knowledge-Based Systems **122**, 167–179 (2017)

DPM Session 3: Use Cases & Privacy Assessment

Dynamic k-anonymity for Electronic Health Records: A Topological Framework

Arjhun Swaminathan[1,2] and Mete Akgün[1,2]

[1] Medical Data Privacy and Privacy Preserving Machine Learning, Department of Computer Science, University of Tübingen, Tübingen, Germany
[2] Institute for Bioinformatics and Medical Informatics, Tübingen, Germany
arjhun.swaminathan@uni-tuebingen.de, mete.akguen@uni-tuebingen.de

Abstract. With the rapid digitization of Electronic Health Records (EHRs), fast and adaptive data anonymization methods have become increasingly important. While tools from topological data analysis (TDA) have been proposed to anonymize static datasets—allowing the creation of multiple generalizations for different anonymization needs from a single computation—the application to dynamic datasets remains unexplored. To address this, our work adapts existing methodologies to the dynamic setting. We develop an improved version of weighted persistence barcodes that track higher-dimensional holes in data, allowing us to edit persistence information on the fly. Additionally, we introduce filtration trimming, a novel technique designed to update persistence data quickly with minimal computing effort when data is added. Our work represents a significant advancement in healthcare data privacy, offering a refined approach to protecting highly sensitive and evolving patient data through dynamic k-anonymity.

Keywords: Persistence Barcodes · k-anonymity · Dynamic datasets · Medical Data Privacy

1 Introduction

In the era of digital transformation, the healthcare sector has accumulated vast amounts of patient data through EHRs, encompassing demographics, medications, diagnoses, progress notes, and medical history. By 2019, 96% of general practitioners in the EU had adopted EHRs[1]. This shift coincides with strict privacy regulations like the GDPR[2], which mandate de-identification or anonymization of sensitive data before processing.

Healthcare providers face the challenge of utilizing this rich data for improved healthcare while ensuring patient privacy. Anonymizing EHRs before public or third-party access enables researchers to conduct large-scale studies, aid in public health monitoring and develop crucial medical treatments without compromising

[1] eHealth adoption in primary healthcare in the EU is on the rise, European Commission
[2] General Data Protection Regulation ((EU) 2016/679)

privacy. The primary technique for anonymizing EHRs has been k-anonymity [30, 37], but it is vulnerable to attacks like background knowledge attacks [22, 34], attribute disclosure attacks [21], membership inference attacks, and linkage attacks [18], as highlighted by Sweeney's analysis of the 1990 U.S. Census data [36]. To address these limitations, l-diversity [22, 45] and t-closeness [21] were developed to enhance k-anonymity. Despite its computational challenges and NP-hardness [25], k-anonymity remains widely used [28, 2, 38, 40, 42, 41, 24].

Differential privacy, another key privacy technique, adds noise to data to ensure that the omission of a single entry does not significantly alter analysis results [11]. It protects against linkage and dataset reconstruction attacks [12] and is used in various fields, such as privacy-preserving data sharing in GWAS studies [14, 1] and health recommender systems [39]. However, differential privacy is analysis-sensitive — the amount of noise added to the data depends on the type of queries or computations executed on it. k-anonymity often emerges as a more general alternative for various applications.

Amongst numerous techniques that achieve k-anonymity, [33] stands out, being the only topologically informed method. It is particularly notable for its unique ability to compute multiple generalizations for different k values without the need to rerun the algorithm for each generalization or each k. This in turn allows for stronger privacy measures such as l-diversity and t-closeness to be applied. This remarkable efficiency is achieved through a single computation of a weighted persistence barcode, using scalable algorithms [23, 3, 9]. However, the method has a critical limitation - it is limited to static datasets. This is a significant drawback in time sensitive fields such as healthcare where data is frequently republished, often with alterations. When handling the dynamic nature of data in such cases, [33] would require a complete re-computation of the weighted persistence barcode for the dataset, leading to significant computational strain.

1.1 Our Contribution and Paper Structure

In our work, we address the major limitation identified in [33] by developing a method that removes the need for complete re-computation of persistent homology when the data changes. We introduce the concept of *hole-weighted persistence barcodes*, a new approach to track the evolution of higher-dimensional holes in the data. This innovation allows us to manage data removal, addition, and updating in dynamic datasets efficiently.

For data removal and updating, our algorithm changes the existing hole-weighted persistence barcodes directly, thus avoiding the need to recalculate persistent homology with every change in the dataset. When new data is added, we use *filtration trimming*, a novel technique which significantly reduces the number of persistent homology re-computations needed. Our proof of concept experimental evaluation using [3] on simulated data exhibits a significant reduction in the number of re-computations needed for data additions compared to [33].

We chose to evaluate our method against Speranzon et al. [33] because of its pioneering role in topology-informed k-anonymity and its unique advan-

tages. To the best of our knowledge, our work is the first topologically-informed anonymization method for dynamic data.

The remainder of the paper is structured as follows. In Section 2, we review k-anonymity and other preliminaries. In Section 3, we discuss related work, with a focus on [33]. In Section 4, we propose our methodology, proposing hole-weighted persistence barcodes, and inductively describing handling of addition, deletion and updating of data. We conclude in Section 5.

2 Preliminaries

2.1 k-anonymity

When employing k-anonymity, one aims to work with datasets whose attributes are categorized into identifiers, quasi-identifiers, and sensitive data, as described in Table 1.

- **Identifiers:** These are attributes like name or Social Security number that can directly identify an individual. These are typically de-identified before data publishing to protect the privacy of individuals.
- **Quasi-identifiers:** These are attributes $A_1, A_2, \ldots, A_M$ such as age, gender, or zip code that cannot individually identify an individual but can do so when combined together.
- **Sensitive data:** These are attributes like medical conditions or salary, which are sensitive information one intends to keep private.

Name	Admission Date	Age	Blood Pressure	Diagnosis
Maria	02.10.2022	23	121 mm Hg	Anxiety
Priya	05.10.2022	44	97 mm Hg	UTI
Ahmed	03.01.2023	21	95 mm Hg	–
Aiden	05.02.2023	41	100 mm Hg	Asthma

Identifiers Quasi-identifiers Sensitive Data

Table 1: Table illustrating the classification of data attributes into identifiers (to be de-identified prior to publication), quasi-identifiers, and sensitive data.

The table of quasi-identifiers is denoted by $T(A_1, A_2, \ldots, A_M)$ consisting of N rows, where A_i are attributes that can take numeric or categorical value. Then the i^{th} sample's j^{th} quasi-identifier data is denoted as T_{ij}. Another table $\bar{T} = (\bar{A}_1, \bar{A}_2, \ldots, \bar{A}_M)$ consisting of N rows is said to be a generalization $\bar{T}$ of T if for all i, j, $T_{ij} \subset \bar{T}_{ij}$.

Definition 1 (k-anonymity [30]). *Consider a generalization $\bar{T}$ of T. $\bar{T}$ is said to have the* k*-anonymity property if every row in $\bar{T}$ appears at least k times in $\bar{T}$.*

T		
02.10.2022	23	121
05.10.2022	44	97
03.01.2023	21	95
05.02.2023	41	100

$\bar{T}$		
2022	$20-50$	$95-125$
2022	$20-50$	$95-125$
2023	$*1$	$70-100$
2023	$*1$	$70-100$

$\bar{T}^*$		
$****$	$**$	$***$
$****$	$**$	$**$
$****$	$**$	$**$
$****$	$**$	$***$

Table 2: Here, T represents the original data table, $\bar{T}$ is the 2-anonymous generalization of T, while $\bar{T}^*$ is an over-generalization of T and $\bar{T}$.

An illustration of k-anonymity is described in Table 2. The objective of k-anonymity is to transform a given table T with quasi-identifiers into a generalized table that satisfies the k-anonymity property. There exist multiple generalizations that achieve k-anonymity, but we aim to minimize data loss, so the anonymized data is private, but also usable for further analysis. A common first step used to achieve k-anonymity, is mapping data to a metric space, and forming a point cloud with the data. This allows for geometric structures to be defined to create generalizations.

Definition 2 (Embedded Point Cloud [33]). *Consider a table of quasi-identifiers T containing M attributes and N samples. An embedded point cloud $P_T \in \mathbb{R}^M$ is formed by treating each row of T as a point in $\mathbb{R}^M$, such that $P_T = \{p_1, p_2, \ldots, p_N\}$, where p_i corresponds to the i^{th} row of T. This data is generally standardized along every attribute.*

3 Related Work

3.1 k-anonymity in practice

The implementation of k-anonymity involves techniques like data generalization, fragmentation, and microaggregation, each with its own distinct advantages. Generalization can use predefined intervals (hierarchy-based) such as [30, 19] or runtime-determined intervals (recoding-based) like Mondrian [20, 33], and our approach. Data fragmentation separates quasi-identifier data from sensitive information [45], while microaggregation forms clusters with a minimum of k-similar records [10, 31, 32, 24].

k-anonymity is applied in various fields. During COVID-19, contact-tracing apps used Bluetooth Low Energy (BLE) to maintain user privacy [28, 2, 38, 40]. These contract tracing applications anonymize data via encrypted beacon IDs or hash-based representations of user contacts, ensuring users were indistinguishable from at least *k-1* others. In location-based services (LBS), spatial cloaking [15, 16, 26] reduces location accuracy to ensure each request is indistinguishable from *k-1* others, using trusted anonymizers. Other applications include road networks [27], autonomous vehicles [42], crowd-sensing [43], and data publishing for research [41].

The utility of k-anonymity to safeguard dynamic datasets has been explored in various studies, each offering vastly different methodologies to ours limiting direct comparability. The seminal work by [5] established a method for efficiently updating k-anonymized frameworks with the addition of new data entries, although it does not address deletions or modifications. In contrast, m-invariance [46] utilizes counterfeit data to preserve privacy when data is either added or removed. Alternatively, [29] employs micro-aggregation to manage changes including additions, deletions, and updates, which is fundamentally different from generalization-based approaches.

Unique in its application, [33] employs topological information to achieve k-anonymity. This method not only supports the generation of multiple generalizations but also caters to varied k-anonymity requirements in a single computation.

3.2 Topology-based k-anonymity

We now briefly discuss the theoretical framework developed by [33], which enables the application of persistent homology to the k-anonymity problem to derive an optimal generalization for a database with minimal loss. We will discuss application to numerical data, as in the foundational work, and this can be extended to categorical data using *generalization trees*.

Definition 3 (Anonymity Complex [33]). *Given an embedded point cloud P_T derived from a table T of quasi-identifiers, an anonymity complex $\mathcal{C}^\epsilon(P_T)$ is the Cêch complex [17] defined over P_T comprising of simplices [17], each having vertices that correspond to points in P_T. A k-simplex is included in $\mathcal{C}^\epsilon(P_T)$ iff the ϵ-ball neighborhoods around its $k+1$ vertices share at least one common point.*

Definition 4 (k-anonymity Complex [33]). *An anonymity complex $\mathcal{C}^\epsilon(P_T)$ is termed a k-anonymity complex if the following conditions are met:*

1. *$S_l = P_T$, where S_l is an l-simplex with $l \geq k-1$.*
2. *$S_{l_1} \cap S_{l_2} = \emptyset$ for all $l_1 \neq l_2$.*
3. *For every p_i in P_T, p_i belongs to some S_l.*

The smallest ϵ for which $\mathcal{C}^\epsilon(P_T)$ satisfies these conditions is termed the global generalization strategy for parameter k, and is denoted by ϵ_k. Other ϵ that satisfies these conditions are other generalizations. We call the sequence of subcomplexes $\phi \subseteq \mathcal{C}^{\epsilon_1}(P_T) \subseteq \ldots \subseteq \mathcal{C}^{\epsilon_n}(P_T)$, where $\epsilon_i < \epsilon_j$ if $i < j$, a filtration.

Notably, it was demonstrated in [33] that an anonymity complex $\mathcal{C}^{\epsilon_k}(P_T)$ is a k-anonymity complex iff it has trivial homology groups $H_n(\mathcal{C}^{\epsilon_k}(P_T)) = 0$ for all $n > 0$.

By mapping T into an embedded point cloud P_T and constructing a Čech filtration as illustrated in Figure 1, we aim to find the smallest ϵ-value, ϵ_k, that forms a k-anonymity complex $\mathcal{C}^{\epsilon_k}(P_T)$. Simplices S_l in the complex form when their $l+1$ vertices have overlapping ϵ_k-ball neighborhoods. However, these simplices do not represent the anonymized equivalence classes. Instead, equivalence

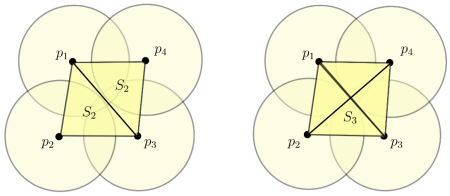


Fig. 1: A comparative visualization of two filtration levels, ϵ_1 and ϵ_2, showcasing the formation of a 3-simplex, S_3, at $\epsilon_2 > \epsilon_1$. While the points do not form a 3-anonymity complex at ϵ_1, they successfully form a 4-anonymity complex at ϵ_2.

classes are defined by computing the circumcenter and circumcircle radius r of each simplex. The equivalence class for a simplex is the higher-dimensional hypercube centered at the circumcenter with side length $2r$. This method generalizes the data, distinguishing equivalence classes from the simplices.

Weighted Persistence Barcodes To identify the optimal generalization strategy ϵ_k for a specific k-anonymity problem, [33] introduced weighted persistence barcodes, as shown in Figure 2. We replicated this using *PHAT* [3]. Unlike traditional persistence barcodes, the weighted versions feature bars with varying thickness, determined by the number of vertices in each H_0 component.

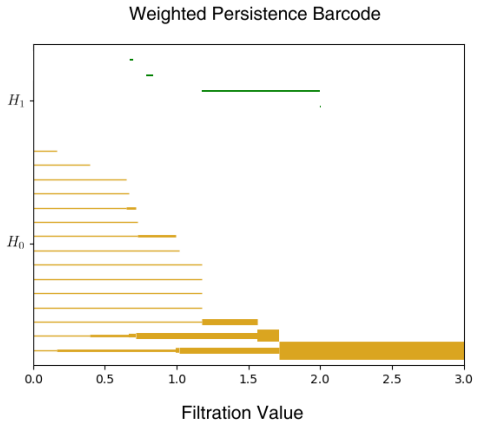


Fig. 2: The weighted persistence barcode describing the merging of components and formation of holes on a simulated dataset consisting of 15 samples and 2 quasi-identifiers.

The goal is to find filtration values where all H_0 bars have a thickness of k, and higher-order homology groups are trivial, indicating a viable k-anonymity complex. These regions are potential generalization values ϵ_k. The number of equivalence classes in a generalization relates to the maximal simplices [17] covering the complex at specific filtration values. Thus, a single persistence barcode calculation for a dataset can identify multiple generalizations for any k, which other algorithms cannot achieve [30, 19, 20, 45, 10, 31, 32, 24, 29].

The smallest viable generalization is optimal, and for stronger properties like l-diversity and t-closeness, other generalizations can be used.

Computational Constraints Although [33] stands out amongst other methods of anonymization with its unique advantages, it comes at a cost since computing persistent homology is a costly operation. When limited to static datasets, the method needs to recompute the persistent homology for the entire data when any change occurs in the dataset. In the worst case scenario, this comes at a computational cost of $\mathcal{O}(\sum_{i}^{M}({}^{N}C_i)^3)$ where N denotes the number of points, and M the number of quasi-identifiers [13, 6, 8].

4 Proposed Method

In this section, we introduce our methodology to utilize persistence information to anonymize dynamic datasets. Recalculating persistent homology with every dataset change is computationally impractical. Our goal is to efficiently use existing persistence barcode information and extract insights as the data evolves. We address data removal, addition, and updating scenarios inductively.

First, we introduce hole-weighted persistence barcodes that help avoid recomputing persistent homology when data is removed. Next, we introduce *filtration trimming*, a technique that significantly reduces the number of persistent homology recomputations made when additional data is added. Finally we discuss data updates, where the stability of persistence diagrams help avoid recomputations.

4.1 Hole-weighted Persistence Barcodes

To capture information about holes across the filtration, we enhance weighted persistence barcodes from [33] to include both component and cycle information. This non-trivial task is detailed in Algorithm 1, which tracks the birth and progression of topological holes. Specifically, we track a k-dimensional hole from its birth ϵ_{birth} to its death ϵ_{death}. We do this by constructing a graph of simplices and using breadth-first search (BFS) [4] to find loops. Persistence data $\mathcal{P}$ will represent simplex formations and corresponding filtration values.

Figure 3 simulated using persistence information with the help of [3] demonstrates that in contrast to the growing thickness of bars in H_0, bars for holes decrease in weight over time due to the ongoing filling of these holes.

We now establish our notation to describe hole-weighted persistence barcodes as visualized in Figure 4. Let's represent each component in H_n as ${}_nB^i$. Each

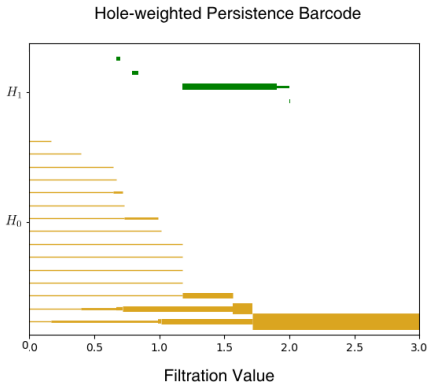


Fig. 3: Hole-weighted persistence barcode describing the merging of components and evolution of holes demonstrated on simulated data comprising 15 samples and 2 quasi-identifiers.

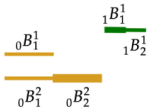


Fig. 4: Visual representation of the notation used.

component ${}_nB^i$ comprises a series of bars represented as $({}_nB_1^i, {}_nB_2^i, \ldots, {}_nB_{n_i}^i)$. Here, ${}_nB_j^i$ denotes the j^{th} bar in the component ${}_nB^i$. n_i represents the number of bars in the i^{th} n-dimensional component.

Further deconstructing, each bar, ${}_nB_j^i$, is represented as a tuple: ${}_nB_j^i = ({}_nb_j^i, {}_nd_j^i, {}_nw_j^i, {}_nV_j^i)$, where:

- ${}_nb_j^i$ denotes the birth of a topological hole.
- ${}_nd_j^i$ denotes the death of a topological hole.
- ${}_nw_j^i$ represents the number of vertices forming the hole.
- ${}_nV_j^i$ is the set of vertices that form the hole.

To ensure chronological coherence, we maintain that for every $j < n$, the relation ${}_nd_j^i = {}_nb_{j+1}^i$ holds true. This alignment presents a sequential understanding of the evolution of topological holes over time.

4.2 Data Removal

Healthcare data management is subject to strict regulations, often requiring time-bound consent for patient information. As a result, healthcare institutions periodically delete patient data from their databases. To handle this removal of data efficiently, we directly modify hole-weighted persistence barcodes instead of recomputing persistent homology across the filtration of the updated data. Without loss of generality, we detail the removal of vertex v_r from the data in Algorithm 2. It is worth noting that we wouldn't need to track holes for further updates in the data since the algorithm inherently produces the hole-weighted persistence barcode associated with the updated data.

Algorithm 1 Tracking k-dimensional holes in a filtration

Require: Persistence data $\mathcal{P}$, ϵ_{birth}, ϵ_{death}.
Ensure: Updated hole-weighted persistence barcode $\mathcal{B}$.
Let $V_I \leftarrow [v_1, v_2, \ldots, v_{k+1}]$ be the simplex forming at ϵ_{birth}.
Initialize $S_k \leftarrow$ all k-simplices from $\mathcal{P}$ formed at or before ϵ_{birth}.
Initialize an undirected graph $G(V, E)$ where $V = S_k$.
for each pair of simplices $s_1, s_2 \in S_k$ **do**
  if $s_1 \cap s_2$ has k vertices **then**
    Add edge (s_1, s_2) to E.
  end if
end for
$G' \leftarrow$ component of G containing V_I.
$L \leftarrow$ BFS(G') to identify all loops.
for each loop $l \in L$ **do**
  $\Delta_l \leftarrow \bigcup_{s \in l} s$
  if NOT $\exists s \in \mathcal{P}$ such that s formed before ϵ_{birth} and $s = \Delta_l$ **then**
    Mark Δ_l as a hole.
  end if
end for
for each k-simplex s formed after ϵ_{birth} and before ϵ_{death} **do**
  if $s \subseteq \Delta_l$ **then**
    Add s as a vertex to G'.
    for each $s_{G'} \in G'$ **do**
      if $s \cap s_{G'}$ has k vertices **then**
        Add edge $(s, s_{G'})$ to G'.
      end if
    end for
    $L' \leftarrow$ loops in Δ_L using BFS(G') such that $s \not\subseteq \Delta_{L'}$.
    Update Δ_L with $\Delta_{L'}$.
  end if
end for
return $\mathcal{B}$ updated based on changes to Δ_L.

Algorithmic Complexity In the worst-case scenario, where the dataset forms a fully connected graph, the BFS algorithm employed for generating the hole-weighted persistence barcode incurs a computational complexity of $\mathcal{O}(\sum_i^M ({}^N C_i)^3)$, where N represents the number of data points and M the dimension of the space the points reside in. This complexity is analogous to that of computing persistent homology for the dataset as discussed earlier in Section 3. Consequently, the cumulative complexity for both computing the persistence information and generating the hole-weighted persistence barcode is $\mathcal{O}(2 \sum_i^M ({}^N C_i)^3)$, in comparison to the $\mathcal{O}(\sum_i^M ({}^N C_i)^3)$ required solely for computing persistence information.

However, an important distinction arises in the context of data removal. While recalculating persistent homology after each data point removal remains computationally intensive, modifying the existing hole-weighted persistence barcode is considerably more efficient, with a complexity of only $\mathcal{O}(N)$. After under-

Algorithm 2 Data removal

Require: Persistence data $\mathcal{P}$, vertex v_r to be removed.
Ensure: Updated hole-weighted persistence barcode.
Collect all ${}_0B^r$ where ${}_0d_1^r$ is the shortest distance between v_r and any other point.
Remove any one component ${}_0B^{\hat{r}}$ that contains only one bar ${}_0B_1^{\hat{r}}$.
if ${}_0V_1^{\hat{r}} = [v_l] \neq [v_r]$ **then**
  for ${}_0B^{\tilde{r}}$ with ${}_0V_1^{\tilde{r}} = [v_r]$ **do**
    Replace every occurrence of v_r with v_l and vice versa.
  end for
end if
for each bar ${}_0B_j^i$ in every component **do**
  if v_r is a member of ${}_0V_j^i$ **then**
    if 1-simplex $[v_t, v_r]$ forms at b_j^i and $v_t \in {}_0V_j^i$ **then**
      $\mathbf{d} \leftarrow \min\{\|v_r - v_s\| \mid v_s \notin {}_0V_j^i\}$.
      Adjust ${}_0b_j^i$ by adding $\mathbf{d}$.
    end if
    Remove v_r from ${}_0V_j^i$.
    Update ${}_0w_j^i = {}_0w_j^i - 1$.
  end if
end for
for each component ${}_{k\geq 1}B^i$ **do**
  if $v_r \in {}_kV_j^i$ **then**
    Remove ${}_kB_j^i$.
  end if
end for
for each component without bars **do**
  Remove the component.
end for
Relabel the persistence barcode to reflect changes.
return Updated hole-weighted persistence barcode.

going K successive data removal operations, [33] would incur a computational complexity of $\mathcal{O}(\sum_{J=N-K}^{N} \sum_{i}^{M} ({}^{J}C_i)^3)$. In contrast, our approach maintains a more manageable complexity of $\mathcal{O}(2 \sum_{i}^{M} ({}^{N}C_i)^3 + KN)$.

4.3 Data Addition

The continuous flow of new data in the healthcare sector highlights the need to regularly update medical datasets. Regular updates help capture changes in patient profiles and medical treatments, improving diagnostic and therapeutic strategies. However, this continuous addition of data presents a challenge: maintaining the privacy of individuals while ensuring k-anonymity. Each new entry can potentially compromise existing anonymizations.

Addressing the complexities of data addition, our concern is the recalibration of persistence barcodes without the need to compute the persistence information for the entire Cêch filtration. We demonstrate that the addition of a single

point does not demand a comprehensive homology computation in Algorithm 3. Instead, updates primarily occur in the local vicinity of the added point. By leveraging the persistent homology computations from the static data, we only recompute specific trimmed segments influenced by the new data. This methodology promotes computational efficiency by reducing redundant calculations.

When introducing a new point, v_{N+1}, we first identify the circumcenters of all simplices that include v_{N+1} up to dimension M within its local neighbourhood. Here, M represents the number of quasi-identifiers. Subsequently, we calculate the distances between these circumcenters and v_{N+1}, sorting them in $\delta = \{\delta_1, \delta_2, \ldots, \}$ where, for $i < j$, $\delta_i < \delta_j$.

Algorithm 3 Data addition

Require: δ, Persistence data $\mathcal{P}$.
Ensure: Updated persistence barcode.
Add new component ${}_0B_1^{N+1} = (0, \min(\delta), 1, [v_{N+1}])$.
for any one v_l such that $\|v_l - v_{N+1}\| = \min(\delta)$ **do**
    Find bar ${}_0B_j^l$ such that ${}_0b_j^l \leq \min(\delta) \leq {}_0d_j^l$ where $v_l \in {}_0V_j^l$.
    Set ${}_0B_j^l = ({}_0b_j^l, \min(\delta), {}_0w_j^l, {}_0V_j^l)$
    Set ${}_0B_{j_+}^l = (\min(\delta), {}_0d_j^l, {}_0w_j^l + 1, {}_0V_j^l \cup [v_{N+1}])$
end for
Identify all δ_i with $i > 1$ and compute persistence in ascending order of δ_i.
Determine homology changes as compared to previous persistence information and denote as $\tilde{\delta} = \{\tilde{\delta}_1, \ldots, \tilde{\delta}_k\}$.
for each $0 \leq i < k$ **do**
    Compute persistent homology using the updatable SNF for the filtrations:

$$\mathcal{C}^{\tilde{\delta}_i} \to \mathcal{C}^{\epsilon_i^1} \to \ldots \to \mathcal{C}^{\epsilon_i^k} \to \mathcal{C}^{\tilde{\delta}_{i+1}}$$

    if $H_k(\mathcal{C}^{\epsilon_i^t})$ matches the previous persistence information for any k at ϵ_i^t **then**
        Move to next filtration
    end if
end for
Relabel indices as per component and bar order.
return Updated persistence barcode

Algorithmic Complexity Calculating a simplex's circumcenter is an $\mathcal{O}(t)$ operation using Welzl's recursive algorithm [44], where t is the dimensionality of the simplex. Assuming we have $\bar{T}$ simplices around the added point, calculating the circumcenters of $\bar{T}$ t-dimensional simplices has a complexity of only $\mathcal{O}\left({}^{\bar{T}}C_{\bar{T}/2}(t/2)\right)$. A key insight emerges from this: computing circumcenters around our new data point is considerably less resource-intensive than computing the persistent homology for an entire filtration.

The parameter δ and its reduction $\tilde{\delta}$ has proven instrumental in our algorithmic approach to trimming the filtration length in our Cêch filtration, as shown in Table 3 and Figure 5, generated using [3]. Here we measure the filtration lengths

Data Points	Added Points	Filtration Length	Trimmed Length
10	1	231	19
10	2	298	20
10	5	575	45
20	1	1561	347
20	5	2625	386
20	10	4525	1115
50	1	22151	3301
50	5	27775	3792
50	10	36050	3374
100	1	171801	15379
100	5	193025	17263
100	10	221925	18760
100	25	325625	19242

Table 3: Filtration Lengths and Trimmed Filtration Lengths for Simulated Data with 2 Quasi-identifiers.

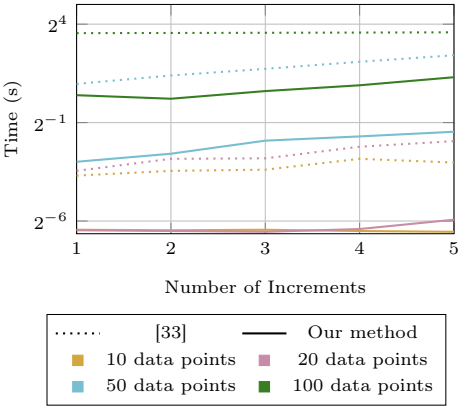


Fig. 5: Comparison of methods when data points are increased by 10% of the original dataset at each step. The time required to compute persistent homology on full and trimmed filtration lengths is plotted.

when additional data is added, and after we perform filtration trimming. The runtimes reported are total persistent homology computation times for the respective filtration lengths. This streamlined process, even with the addition of extra data points, dramatically cuts down on computational overhead, especially when considering larger datasets. For our experiments, we worked with data simulated using two quasi-identifiers to maintain simplicity.

4.4 Data Updatability

In rare cases when EHR data entries are modified, whether to refine diagnoses or correct data, it raises challenges for maintaining k-anonymity.

Based on the findings from [7], minor data perturbations have minimal impact on persistent homology. This suggests a practical approach: for slight data modifications, first check if the current anonymized dataset meets k-anonymity standards. If not, manage changes using protocols for data removal and addition. If the data is minimally altered and k-anonymity is satisfied, updating the hole-weighted persistence barcode is straightforward. If a component or hole arises due to the formation of a simplex $[\ldots, v_u, \ldots]$ with v_u being the adjusted data, we modify the associated bar by computing the circumcenter of the simplex.

4.5 Comparison with related literature

Below, in Table 4, we present a comprehensive comparison between our method and established k-anonymity techniques against key criteria. These were chosen to highlight the strengths and potential areas for improvement of our approach

relative to the current state-of-the-art. Our work, and [33] stand out as the only two topology-informed methods, that can generate multiple generalizations for multiple anonymization criteria with a single computation. Additionally, our method's ability to handle dynamic data allows us to adapt not only to changing data but also to evolving privacy requirements.

Method	Technique Applied	Handles Categorical Data	Multiple Generalisations	Multiple Anonymizations	Adaptability to Dynamic Databases
[20]	Recoding-based	×	×	×	×
[19]	Recoding/Hierarchy-based	✓	×	×	×
[35]	Recoding/Hierarchy-based	✓	×	×	×
[5]	Recoding-based	×	×	×	partially
[46]	Recoding/Hierarchy-based	✓	×	×	✓
[29]	Microaggregation	✓	×	×	✓
[33]	Recoding-based	×	✓	✓	×
Our Method	Recoding-based	×	✓	✓	✓

Table 4: Comparison of various k-anonymity methods.

5 Conclusion and Future Work

In conclusion, our work substantially advances the application of TDA techniques for privacy-preserving dynamic data publishing in the context of EHRs. Previous work in the field focused on static datasets [33] and required computation of persistent homology on the whole dataset whenever data is updated. Our methodology extends this by addressing key challenges in dynamic databases, including data removal, addition, and updatability, enabling rapid adaptability without the need for extensive recomputation of persistent homology for updated data. Our solution can adapt not only to dynamic data, but dynamic privacy demands without requiring additional computational strain.

For data removal, we have innovatively utilized hole-weighted persistence barcodes, constructed through a breadth-first search algorithm on a graph derived from existing persistence data. This approach allows for systematic editing of the barcodes upon data removal. In the case of data addition, we have refined the process by trimming the Cêch filtration, thus significantly reducing the number of persistent homology recomputations required, as our experiments have demonstrated. Regarding data updatability, we focused on the stability of persistence diagrams and computed a limited number of simplex circumcenters, avoiding expensive persistent homology re-computations.

Our approach enhances the balance between data utility, anonymization flexibility and patient privacy, which is crucial in the context of rapid EHR adoption

and stringent regulations like the GDPR. Although our work primarily addresses numerical data, extending these methodologies to include categorical data using zigzag persistent homology is a promising direction for future research. Moreover, incorporating more robust privacy measures such as l-diversity and t-closeness could enhance the algorithm. As tools for persistence computations continue to evolve, the scalability and practical applicability of our approach are expected to increase, providing greater utility across various applications.

Availability of software code

Our code to implement hole-weighted persistence barcodes is available at the URL: https://github.com/mdppml/dynamic-topological-k-anonymity.git

Acknowledgments

This research was supported by the German Ministry of Research and Education (BMBF), project number 01ZZ2010. We thank Saradha Senthil Velu for their invaluable discussions that helped in formulating the ideas presented in the paper.

References

1. Akgün, M., Bayrak, A.O., Ozer, B., Sağıroğlu, M.Ş.: Privacy preserving processing of genomic data: A survey. Journal of biomedical informatics **56**, 103–111 (2015)
2. Ali, J., Dyo, V.: Cross hashing: Anonymizing encounters in decentralised contact tracing protocols. In: 2021 International Conference on Information Networking (ICOIN). pp. 181–185. IEEE (2021)
3. Bauer, U., Kerber, M., Reininghaus, J., Wagner, H.: Phat–persistent homology algorithms toolbox. Journal of symbolic computation **78**, 76–90 (2017)
4. Bundy, A., Wallen, L.: Breadth-first search. Catalogue of artificial intelligence tools pp. 13–13 (1984)
5. Byun, J.W., Sohn, Y., Bertino, E., Li, N.: Secure anonymization for incremental datasets. In: Secure Data Management: Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006. Proceedings 3. pp. 48–63. Springer (2006)
6. Carlsson, G., De Silva, V., Morozov, D.: Zigzag persistent homology and real-valued functions. In: Proceedings of the twenty-fifth annual symposium on Computational geometry. pp. 247–256 (2009)
7. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Stability of persistence diagrams. In: Proceedings of the twenty-first annual symposium on Computational geometry. pp. 263–271 (2005)
8. Cohen-Steiner, D., Edelsbrunner, H., Harer, J.: Extending persistence using poincaré and lefschetz duality. Foundations of Computational Mathematics **9**(1), 79–103 (2009)
9. Cohen-Steiner, D., Edelsbrunner, H., Morozov, D.: Vines and vineyards by updating persistence in linear time. In: Proceedings of the twenty-second annual symposium on Computational geometry. pp. 119–126 (2006)
10. Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogeneous k-anonymity through microaggregation. Data Mining and Knowledge Discovery **11**, 195–212 (2005)

11. Dwork, C.: Differential privacy. In: International colloquium on automata, languages, and programming. pp. 1–12. Springer (2006)
12. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Foundations and Trends® in Theoretical Computer Science **9**(3–4), 211–407 (2014)
13. Edelsbrunner, Letscher, Zomorodian: Topological persistence and simplification. Discrete & Computational Geometry **28**, 511–533 (2002)
14. Fienberg, S.E., Slavkovic, A., Uhler, C.: Privacy preserving gwas data sharing. In: 2011 IEEE 11th International Conference on Data Mining Workshops. pp. 628–635. IEEE (2011)
15. Gedik, B., Liu, L.: Protecting location privacy with personalized k-anonymity: Architecture and algorithms. IEEE Transactions on Mobile Computing **7**(1), 1–18 (2007)
16. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: Proceedings of the 1st international conference on Mobile systems, applications and services. pp. 31–42 (2003)
17. Hatcher, A.: Algebraic topology. (2005)
18. Kitamura, K., Irvan, M., Shigetomi Yamaguchi, R.: Disclosure of multiple" patient characteristics" format statistics leaks quasi-identifier linkage. In: Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics. pp. 15–25 (2023)
19. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proceedings of the 2005 ACM SIGMOD international conference on Management of data. pp. 49–60 (2005)
20. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: 22nd International conference on data engineering (ICDE'06). pp. 25–25. IEEE (2006)
21. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: 2007 IEEE 23rd international conference on data engineering. pp. 106–115. IEEE (2006)
22. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) **1**(1), 3–es (2007)
23. Maria, C., Boissonnat, J.D., Glisse, M., Yvinec, M.: The gudhi library: Simplicial complexes and persistent homology. In: Mathematical Software–ICMS 2014: 4th International Congress, Seoul, South Korea, August 5-9, 2014. Proceedings 4. pp. 167–174. Springer (2014)
24. Mayer, R., Karlowicz, A., Hittmeir, M.: K-anonymity on metagenomic features in microbiome databases. In: Proceedings of the 18th International Conference on Availability, Reliability and Security. pp. 1–11 (2023)
25. Meyerson, A., Williams, R.: On the complexity of optimal k-anonymity. In: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 223–228 (2004)
26. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: Query processing for location services without compromising privacy. In: VLDB. vol. 6, pp. 763–774 (2006)
27. Mouratidis, K., Yiu, M.L.: Anonymous query processing in road networks. IEEE Transactions on Knowledge and Data Engineering **22**(1), 2–15 (2009)
28. Park, J., Ahmed, E., Asif, H., Vaidya, J., Singh, V.: Privacy attitudes and covid symptom tracking apps: Understanding active boundary management by users. In: International Conference on Information. pp. 332–346. Springer (2022)

29. Salas, J., Torra, V.: A general algorithm for k-anonymity on dynamic databases. In: Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2018 International Workshops, DPM 2018 and CBT 2018, Barcelona, Spain, September 6-7, 2018, Proceedings 13. pp. 407–414. Springer (2018)
30. Samarati, P.: Protecting respondents identities in microdata release. IEEE transactions on Knowledge and Data Engineering **13**(6), 1010–1027 (2001)
31. Soria-Comas, J., Domingo-Ferrer, J.: Probabilistic k-anonymity through microaggregation and data swapping. In: 2012 IEEE International Conference on Fuzzy Systems. pp. 1–8. IEEE (2012)
32. Soria-Comas, J., Domingo-Ferrer, J., Sánchez, D., Martínez, S.: Enhancing data utility in differential privacy via microaggregation-based k-anonymity. The VLDB Journal **23**(5), 771–794 (2014)
33. Speranzon, A., Bopardikar, S.D.: An algebraic topological perspective to privacy. In: 2016 American Control Conference (ACC). pp. 2086–2091. IEEE (2016)
34. Sun, Y., Yin, L., Liu, L., Xin, S.: Toward inference attacks for k-anonymity. Personal and ubiquitous computing **18**, 1871–1880 (2014)
35. Sweeney, L.: Guaranteeing anonymity when sharing medical data, the datafly system. In: Proceedings of the AMIA Annual Fall Symposium. p. 51. American Medical Informatics Association (1997)
36. Sweeney, L.: Simple demographics often identify people uniquely. Health (San Francisco) **671**(2000), 1–34 (2000)
37. Sweeney, L.: Achieving k-anonymity privacy protection using generalization and suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(05), 571–588 (2002)
38. Tedeschi, P., Bakiras, S., Di Pietro, R.: Iotrace: a flexible, efficient, and privacy-preserving iot-enabled architecture for contact tracing. IEEE Communications Magazine **59**(6), 82–88 (2021)
39. Valdez, A.C., Ziefle, M.: The users' perspective on the privacy-utility trade-offs in health recommender systems. International Journal of Human-Computer Studies **121**, 108–121 (2019)
40. Vaudenay, S.: Analysis of dp3t-between scylla and charybdis (2020)
41. Verdonck, J., De Boeck, K., Willocx, M., Lapon, J., Naessens, V.: A hybrid anonymization pipeline to improve the privacy-utility balance in sensitive datasets for ml purposes. In: Proceedings of the 18th International Conference on Availability, Reliability and Security. pp. 1–11 (2023)
42. Wang, J., Cai, Z., Yu, J.: Achieving personalized k-anonymity-based content privacy for autonomous vehicles in cps. IEEE Transactions on Industrial Informatics **16**(6), 4242–4251 (2019)
43. Wang, X., Liu, Z., Tian, X., Gan, X., Guan, Y., Wang, X.: Incentivizing crowdsensing with location-privacy preserving. IEEE Transactions on Wireless Communications **16**(10), 6940–6952 (2017)
44. Welzl, E.: Smallest enclosing disks (balls and ellipsoids). In: New Results and New Trends in Computer Science: Graz, Austria, June 20–21, 1991 Proceedings. pp. 359–370. Springer (2005)
45. Xiao, X., Tao, Y.: Anatomy: Simple and effective privacy preservation. In: Proceedings of the 32nd international conference on Very large data bases. pp. 139–150 (2006)
46. Xiao, X., Tao, Y.: M-invariance: towards privacy preserving re-publication of dynamic datasets. In: Proceedings of the 2007 ACM SIGMOD international conference on Management of data. pp. 689–700 (2007)

Using Static Code Analysis for GDPR Compliance Checks

Andreas M. Binder[1][0009−0001−0718−977X] and Immanuel Kunz[2][0000−0002−4669−0030]

[1] Fraunhofer AISEC, Garching b. München, Germany
andreas.binder@aisec.fraunhofer.de
[2] Fraunhofer AISEC, Garching b. München, Germany
immanuel.kunz@aisec.fraunhofer.de

Abstract. Ensuring compliance with the General Data Protection Regulation (GDPR) remains a labor-intensive activity, especially in large applications. Moreover, legal experts often do not have the technical knowledge to assess source code. Privacy threat modeling can be used to systematically guide the assessment of privacy threats in designs and code, but it is time-intensive and needs to be redone for changes. In this paper, we build on an existing approach to automate privacy threat modelling using static code analysis and extend it for GDPR compliance checks. We first derive code properties from individual GDPR articles, implement them in a static code analysis tool, and propose queries for the automated analysis of source code. Finally, we evaluate the results using a novel test suite.

Keywords: Static Code Analysis · Privacy Threat Modeling · GDPR Compliance Engineering.

1 Introduction

The General Data Protection Regulation (GDPR) [6] of the European Union has established a high bar for data protection in software products that process personal data. Due to lacking knowledge about the legal texts and their implications, implementing these legal requirements remains a challenge for software engineers. Especially in large software systems with ongoing changes, ensuring compliance to data protection regulations is a difficult task.

There are numerous approaches for implementing privacy-friendly software, like checklists [11,1], Privacy Design Strategies [10] and Privacy Design Patterns [2], but approaches to automatically verify compliance with data protection regulations, like the GDPR, are currently missing. Privacy threat modeling methods such as LINDDUN [5,15] can uncover non-compliance issues using a data flow diagram, but they focus on basic GDPR principles like data minimization and do not cover specific articles of the GDPR. Creating a data flow diagram and performing threat modeling require a comprehensive understanding of the system and significant effort, and errors can affect the accuracy and reliability of the

process. Additionally, threat modeling is a manual and time-consuming process, making it difficult to integrate into the short sprints of agile software development [8]. In comparison, automated approaches realized through static code analysis offer an effective way to reduce effort and can be well integrated into agile software development cycles. However, existing tools cannot check source code for compliance with individual articles of the GDPR. For example, there are tools that attempt to uncover data flows of personal data [3,14] or that check source code for compliance with an individually defined privacy policy [7,13]. As existing tools do not offer the possibility to check source code for compliance with indvidual articles of the GDPR, manual assessment is still being used. As a result, legal experts, who may not have the necessary expertise in understanding source code and software architectures, are required to support the development process and ensure that the software complies with legal requirements.

In this paper, we present a tool for static code analysis, which enables a (semi-)automated check of source code for GDPR compliance. With this tool we want to address the time-consuming and error-prone process of verifying compliance of source code with data protection requirements. To this end, we first create a generic example (Section 3), which is used to derive workflows that a service provider can implement to comply with selected articles of the GDPR. In the same section, we analyze the properties of these workflows and derive source code properties from them, for example data flows and database operations. Based on the resulting properties, in Section 4 an existing static code analysis tool (the Privacy Property Graph (PPG) [12]) is extended to allow mapping of all necessary code properties on the graph. Subsequently, compliance checks are developed using reusable queries in the Cypher query language that enable automated verification of the source code via the PPG. In summary, the following contributions are presented:

- A translation of GDPR articles to low-level code properties that can be identified in a static code analysis
- An extension of an existing static code analysis tool for the automated detection of the identified code properties, in the form of a code property graph
- Reusable queries in the Cypher language to automatically check the graph for indications of (non-)compliance with the GDPR articles.

2 Background and Related Work

Code property graphs (CPGs) are representations of source code that allow the analysis of large source code projects [16]. They include nodes that represent elements of the source code and edges that put the nodes into relation. Properties that CPGs can depict include the program's syntactic structure, its control flow, data flows, as well as program dependencies. In the context of privacy analysis, the representation of data flows in a Data Flow Graph (DFG) is essential to track the flow of personal data. The resulting graph can be stored in a graph database and can then be queried using a query language manually or automatically. This way, problematic data usage patterns can be uncovered.

Banse et al. [4] have extended a CPG library for the analysis of distributed cloud applications. Kunz et al. [12], have then built on this work to also cover the analysis of privacy threats, called the Privacy Property Graph (PPG). The PPG focuses on the automatic detection of LINDDUN threats [5]. Yet, LINDDUN does not directly address compliance with individual GDPR articles but only addresses high-level data protection principles, like data transparency.

The Privacy Flow Graph [14] focuses on visualizing personal data flows to support, e.g., privacy impact assessments. It analyzes source code to uncover data flows of personal data, presenting them in a graphical format. The graphical representation of the data flows can help auditors to uncover problematic data flows. An automated GDPR compliance check, however, is not addressed.

Hjerppe et al. [9] use an Abstract Syntax Tree to track personal data via annotations made by developers in code to automate the creation of a privacy policy. Through the help of annotations the authors pointed out that it is easier to spot personal data processing and storage. Their aim is to support the documentation of personal data processing and facilitate the development of tooling as we propose it in this paper.

Ferrara et al. [7] also explore static code analysis for GDPR checks, but focus on detecting data leaks. Their approach is based on custom policies that define allowed data flows. Commercial tools include Privado [13], which is partly open-source and also uses custom rules to define (non-)compliant program behavior.

Each approach facilitates GDPR compliance checks but a considerable gap remains in the automated code-level check of specific GDPR articles. In this paper we build on the PPG as it already addresses privacy threats on code-level, is open source, and easily extendable.

3 Approach: Extraction of Code Properties

In this section we present our approach for automated GDPR compliance checks. We derive code properties from GDPR articles, which can be detected through static code analysis. Since the articles of the GDPR are defined in an abstract manner, it can be challenging to translate them to the implementation-level. To bridge the gap between the abstract level and concrete implementation suggestions, we introduce a generic example service. This service is designed as a typical client-server architecture, making it a representative model for many types of services that process personal data and are subject to the GDPR.

A complete analysis of the GDPR is not in scope of this paper and is not meaningful as many articles specify authority competences (Art. 51-59), remedies, liabilities, and penalties (Art. 77-84), as well as other provisions that are not reflected in source code. In this paper, we focus on articles that often imply direct interaction of users with their data, i.e. access (Art. 15), rectification (Art. 16), erasure (Art. 17), and portability (Art. 20). Note, however, that the PPG already largely covers Art. 15 which is why we focus on Articles 16, 17 and 20 in the following. We expect that future work can build on the results to verify com-

pliance with many other articles, such as identifying automated decision-making (Art. 22) and determining flows to geolocations outside the EU (Art. 44).

Running example: The online notepad The example depicted in Figure 1 is an online notepad where registered users can store and retrieve personal notes. Through a web app, users can interact with the online notepad and view all their data, such as notes and account information, which is stored in the database. The management server receives requests from users, converts them into database queries, and sends the results back to the web app. Additionally, the management server sends headlines of the notes and user interaction frequencies to an external provider for advertising purposes. This example reflects a commonly used client-server architecture, which we use in the following to derive code properties. All communication between the parties is assumed to be handled via REST interfaces and the HTTP protocol. Furthermore, all communication to external parties is assumed to be done via the management server, i.e. from the data controller. The limitations of this assumption including reliance on the HTTP protocol for general applicability are discussed in Section 5.4.

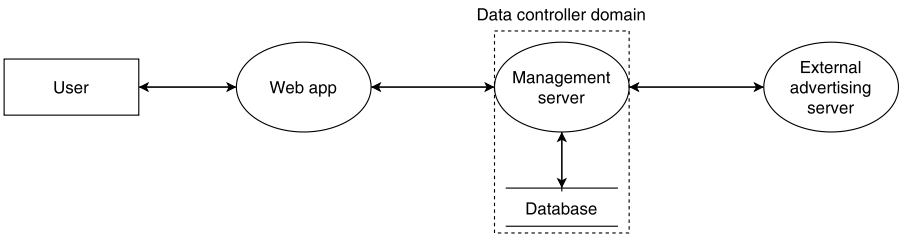


Fig. 1. Data flow diagram of the running example: The *User* entity can interact with the *Web app* provided by the data controller, which in turn communicates with a *Management server*. This server processes the incoming data and may use the *Database* to store, retrieve or delete data. The *Management server* and the *Database* are representing the domain of the data controller, which shares data with a third party's advertising server (*External advertising server*). All data flows between the elements are realized with the HTTP protocol.

In the following we analyze Articles 16, 17 and 20 in the context of the running example to derive code properties.

3.1 Article 16 - Right to Rectification

Article 16 states that the data subject has the right to request rectification of his personal data by the data controller and to complete personal data that is incomplete with additional data.

For the running example, the online notepad service can adopt this workflow to comply with Article 16:

1. The user can edit or complete his personal data within the *Web app* via a user interface. The changes are then passed to a function linked to the user interface.
2. After changing personal data, the *Web app* communicates these changes to a specified REST interface of the *Management server* using a PUT HTTP call (e.g. via URL path `/user/user_id`).
3. The *Management server* processes the incoming data and initiates the re-save by forwarding the user's rectified data to the *Database* via an update database query.

The following **code properties** can be derived:

1. **Prop-data-flow**: A directed, chronological data flow for each personal datum, starting from the property marked as *(*Entry point)* to the component labeled *(*Exit point)*
2. **Prop-ui-editing-form**: An editing form in the client's code that allows editing personal data *(*Entry point)*.
3. **Prop-put-http-request**: A PUT `HttpRequest` in the client's code that addresses an `HttpEndpoint` specified on the server.
4. **Prop-put-http-endpoint**: An `HttpEndpoint` in the server's code, which is addressed by the client program's HTTP PUT request.
5. **Prop-update-database-operation**: A `DatabaseOperation` in the server's code that performs an update of already stored data *(*Exit point)*.

3.2 Article 17 - Right to Erasure

Paragraph 1 Article 17(1) states that the data subject has the right to request an erasure of his personal data by the data controller. The grounds can be, for example, a revocation of consent to the processing or the unlawful processing of data.

For the running example, the online notepad service can adopt this workflow to comply with Article 17(1):

1. The user requests the deletion (e.g. of a personal note) within the *Web app* via an UI-element, e.g. a button, which triggers the function described in the next steps.
2. The *Web app* communicates the deletion request to a specified REST interface of the *Management server* using a HTTP call of type DELETE, e.g. using the URL path `/user/user_id/notes/note_id`
3. The *Management server* processes the incoming request and initiates the deletion of the corresponding database entry via a delete query.

Paragraph 2 The second paragraph states that if the data controller discloses personal data to a further data processor and an erasure of the data has taken place in accordance with paragraph 1, the other data processors shall be informed that the data subject has requested an erasure.

For the running example, the online notepad service can adopt this workflow to comply with Article 17(2):

1. The user requests deletion (e.g. of a personal note) from the *Web app* via an UI-Element, e.g. a button, which invokes a function call, that triggers the process described in the next steps.
2. The *Web app* communicates the deletion request to a specified REST interface of the *Management server* using an HTTP call of type DELETE, e.g. using the URL path `/user/user_id/notes/note_id`.
3. The *Management server* informs all other data processors who have received personal data of the user (in the case of the running example the *External advertising server* received headlines of the notes). This is realized via an HTTP call of type DELETE, which is called from the *Management server*.

Derived Code Properties Since the paragraphs 1 and 2 of Article 17 are closely related to each other, the code properties are nearly the same, except one difference, which can be seen in the following derived **code properties**:

1. **Prop-data-flow**
2. **Prop-ui-button-linked-function** *(*Entry point)*
3. **Prop-delete-http-request**
4. **Prop-delete-http-endpoint**
5. (only for Article 17(1)): **(Prop-delete-database-operation)**: A `Database-Operation` in the server's code that performs a deletion of the stored data *(*Exit point)*.
6. (only for Article 17(2)): **Prop-delete-http-request-extern**: A DELETE `HttpRequest` in the server code targets a REST interface not maintained or run by the service provider*(*Exit point)*.

3.3 Article 20 - Right to Data Portability

Paragraph 1 Article 20(1) states that the data subject has the right to receive his personal data in a structured, commonly used, machine-readable format, which allows the transmission of his personal data to another data controller. This right is given to the person if consent to the processing of the data was given and the processing of the data is automated.

The workflow we expect the notepad service provider to implement assumes that consent for data processing has been obtained and the processing is automated. For the running example, the online notepad service can adopt this workflow to comply with Article 20(1):

1. The user can request via a form in the *Web app* that the service provider delivers the user's personal data in a machine-readable format (e.g. CSV or JSON). The form is linked to a function, which initiates the process described in the next steps.
2. The *Web app* communicates this request to a specified REST interface on the *Management server* using an HTTP call of type GET, for example with the URL path `/user/user_id`.

3. The *Management server* queries all the user's personal data, which are stored in the *Database*, using a database read query.
4. The *Management server* passes the user's personal data to the *Web app* by answering the HTTP request.
5. The *Web app* saves the user's personal data to a file that is machine-readable. E.g. the file format is JSON or CSV.

The following **code properties** can be derived:

1. **Prop-data-flow**
2. **Prop-ui-button-linked-function** *(*Entry point)*
3. **Prop-get-http-request**
4. **Prop-get-http-endpoint**
5. **Prop-retrieve-database-operation**: A `DatabaseOperation` in the server's code that retrieves all stored data from the database.
6. **Prop-json-data-format**: The personal data is in a machine-readable format (e.g. JSON).
7. **Prop-data-storage** *(*Exit point)*

Paragraph 2 Article 20(2) states that the data subject has the right to request the data controller to transfer his personal data directly to another data controller.

For the running example, the online notepad service can adopt this workflow to comply with Article 20(2):

1. The user can request the transferral of his personal data in a machine-readable format (e.g. JSON) to another data controller via a form in the *Web app*. The submission of the form triggers a function executing the process described in the next steps.
2. The *Web app* communicates this request to a specified REST interface on the *Management server* using an HTTP call of type GET, for example with the URL path `/user/user_id?destination=example_destination`
3. *Management server* queries all the user's personal data, which are stored in the *Database*, using a database read query.
4. The *Management server* converts all personal data into JSON format.
5. The *Management server* passes the data in a machine-readable format to the specified destination of the user by adressing a REST interface via a POST HTTP call.

The following **code properties** can be derived:

1. **Prop-data-flow**
2. **Prop-ui-button-linked-function** *(*Entry point)*
3. **Prop-get-http-request**
4. **Prop-get-http-endpoint**
5. **Prop-retrieve-database-operation**
6. **Prop-json-data-conversion**
7. **Prop-post-http-request-extern**: A POST HTTP request in the server code targets a REST interface not maintained or run by the service provider *(*Exit point)*.

4 Implementation

Various tools could be used for the implementation of the compliance checks. In this implementation we leverage the PPG [12] which already provides support for many of the code properties identified in Section 3. We thus map out the gaps between the PPG implementation and the code properties identified above and implement them. In Section 4.1 we present the modifications we applied to enrich the graph with the properties of interest. In Section 4.2, we develop reusable queries, which check the compliance of programs to the respective articles from Section 3. Thereby, generic and reusable queries are written, which are applicable across applications and programming languages. The enhancements of the PPG as well as the evaluation test suite (Section 5.1) are published in the PPG open-source repository[3].

4.1 Enhancements of the PPG

The PPG creates the graph by first applying the underlying CPG library to it and it then adds further nodes and edges through dedicated *passes*. Each pass analyzes the source code and the code property graph that already has been created for it (see Section 2), and modifies it further, e.g., to add nodes and edges for HTTP connections or database operations.

To implement the missing code properties, we create such passes or extend already existing ones. The resulting graph is stored in a Neo4j database which provides the SQL-like Cypher language to query the database.

Firstly, the *DatabaseOperationPass* needs to be enhanced to enable the PPG to identify and integrate various database operations into the graph. This involves introducing a new type property for 'DatabaseQuery' nodes to differentiate among query types such as CREATE, READ, UPDATE, DELETE, and UNKNOWN (e.g. the called function executes an arbitrary database query, which cannot be assessed before runtime).

Secondly, for the detection of HTTP requests and endpoints, the *HttpRequestPass* must be extended. This involves creating 'HttpRequest' and 'HttpEndpoint' nodes, with additional capabilities to recognize PUT and DELETE requests and endpoints. An important addition is the 'url' property for 'HttpRequest' nodes for the identification of destination URLs in HTTP requests.

Lastly, a significant enhancement involves the introduction of the *FileWritePass* to detect file write operations. This requires the creation of a new 'FileWrite' class in the CloudPG ontology, representing an abstraction for all function calls that write to a file, regardless of the programming language. This class should link to the respective 'CallExpression' node performing the file write operation. Moreover, language-specific passes are needed for different programming languages to accurately detect file write operations.

[3] Currently open pull request in repository: `https://github.com/clouditor/cloud-property-graph`

4.2 Development of Compliance Checks

Having extended the PPG with passes that add further code properties to the generated graph, we describe in this section how the graph can be used to check for compliance to the respective articles (semi-)automatically. Users of these compliance checks could be developers, designers, auditors, and privacy experts. The queries are designed to be applicable to various software structures and application types.

A key objective in crafting these queries is to minimize false-negative results, i.e. false indications about a program's compliance. However, given the complexity of unambiguously determining compliance, our goal is to design queries to report more false positives (incorrect non-compliance) rather than false negatives (incorrect compliance). This approach ensures that compliance is only indicated when there is a strong likelihood of its veracity. It is better if users who analyze the results filter out false-positive results than to overlook an actual non-compliance that may lead to costly design and implementation changes later (see also Section 5.4).

These queries enable automatic non-compliance checks through the Neo4j API, which interacts directly with the Neo4j database. They can be executed directly on the database. Additionally, these queries can also be used for manual compliance checks using the Neo4j UI.

Preliminary steps To automatically verify if source code complies with the GDPR through static code analysis, personal data must be identifiable. We propose that personal data be annotated in the code with labels, such as `PseudoIdentifier`. This label has to be added to every variable declaration that already contains or will contain personal data when executing the software. This step has to be executed first in order to be able to track personal data flows in the following and subsequently analyzing them.

Article 16 - Right to rectification We assume compliance with Article 16 if a user can modify any of his stored personal data (Section 3.1). We check for every flow of personal data that is stored in a database, if another data flow exists, starting from a PUT HTTP request. This PUT HTTP request must then lead to a PUT HTTP endpoint on the server, which receives the personal data and then leads to a database update query. Also it must be ensured that the database update query is performed at the the same database in which the personal datum was initially stored. We show the developed query, checking for all derived code properties (Section 3.1) in Listing 1.1 and a plot of the data flows described above in Figure 2.

Note that in the Cypher queries we use the term *path* to denote a path of data flows throughout the program. To query the graph and uncover these paths, we use Neo4j's query language Cypher. It uses round brackets to denote nodes and their types ((:Expression)), dashes for undirected edges (–), arrows

for directed edges (–>), and square brackets in between dashes to denote edge types (-[:ANONYMIZES]-). The *-operator denotes a path of arbitrary length[4].

```
MATCH path1=(ps1:PseudoIdentifier)--()-[:DFG*]->(hr1:HttpRequest {name:
    'POST'})-[:TO]->(he1:HttpEndpoint)--()-[:DFG*]->(d1:DatabaseQuery {
    type: 'CREATE'})
WHERE NOT EXISTS
 {MATCH path2=(ps1)--()-[:DFG*]->(hr3:HttpRequest {name: 'PUT'})-[:TO]->(
    he3:HttpEndpoint {method: 'PUT'})--()-[:DFG*]->(d2:DatabaseQuery)
   WHERE (d2.type='UPDATE')/(d2.type='DELETE') AND (d1)-[:STORAGE]->(:
      DatabaseStorage)<-[:STORAGE]-(d2)}
RETURN path1
```

Listing 1.1. Cypher query which identifies non-compliant Article 16 and 17 personal data flows. `path1` traces all tagged personal data in a database. `path2` follows `path1` data leading to an update (Art. 16) or delete (Art. 17) query in the same database. If `path2` is absent, indicating non-compliance, `path1` is returned.

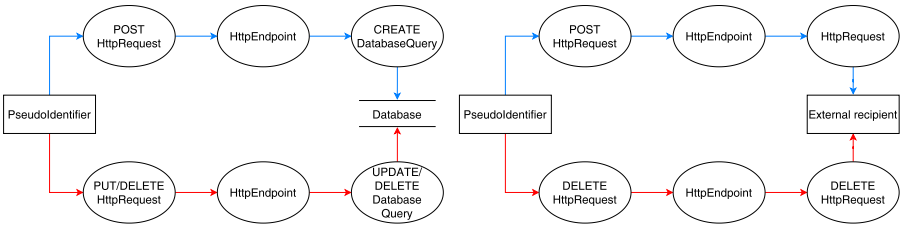


Fig. 2. Diagram consisting of two data flow diagrams – Left data flow diagram: Illustrating `path1` (in blue) and `path2` (in red) data flows for assessing non-compliance with Articles 16 and 17. Right Data flow diagram: Shows two data flows `path1` (blue) and `path2` (red) of the query for the non-compliance check of Article 17(2).

Article 17 - Right to erasure The first paragraph describes that it must be possible for the user to request the deletion of his stored personal data. We check for each flow of personal data to a database, if another data flow exists starting with a HTTP DELETE request. This HTTP DELETE request must lead to an HTTP endpoint at the server, which receives the personal data leading to a database delete operation within the same database in which the personal datum was initially stored. The respective query, which checks for all derived code properties (Section 3.2), can be seen in Listing 1.1. A plot of the data flows described above can be seen in Figure 2.

The second paragraph describes that the data controller informs the other recipients of the personal data as soon as the user requests deletion. We identify all data flows that are communicating personal data to an external party. Note that a communication to an external party can be detected via the PPG because the source code of the external party is not known and therefore the

[4] See https://neo4j.com/developer/cypher/querying

PPG does not create an `HttpEndpoint` node and connects it via a `TO` edge to the `HttpRequest`. Knowing all data flows, communicating personal data to an external data recipient, we check whether the external data receiver is informed about the deletion request. This is the case if another data flow for every personal datum exists starting from HTTP delete request. This HTTP delete request must then lead to an HTTP endpoint on the server that receives the personal data and informs all external recipients of the data about the deletion via an HTTP delete request. We list the developed query, checking for all derived code properties (Section 3.2) in Listing 1.2 and the plot of the data flows described above in Figure 2.

```
MATCH (hr1:HttpRequest), path1=(ps1:PseudoIdentifier)--()-[:DFG*]->(hr1)
WHERE NOT (hr1)-[:TO]-(:HttpEndpoint) AND NOT EXISTS
 {MATCH path2=(ps1)--()-[:DFG*]->(hr2:HttpRequest {name: 'DELETE'})-[:TO
     ]-(he2:HttpEndpoint {method: 'DELETE'})--()-[:DFG*]->(hr3:
     HttpRequest)
    WHERE (hr3.name='DELETE') AND (hr3.url = hr1.url) AND NOT (hr3)-[:TO
        ]-(:HttpEndpoint)}
RETURN path1
```

Listing 1.2. Cypher query detects Article 17(2) non-compliance in data flows. `path1` tracks personal data with `PseudoIdentifier` shared with external parties. `path2` traces notifications to these parties about deletion requests. Absence of `path2` signals non-compliance, returning `path1`.

Article 20 - Right to data portability The first paragraph states that the user has the right to receive his stored personal data in a machine-readable format. We check whether for each personal datum stored in the database a data flow, starting from a HTTP request of type GET exists. This HTTP request then in turn leads to an HTTP endpoint, leading to a database query that loads the personal data from the same database, where it was initially stored and returns it to the client. Returning data is indicated by HTTP status code OK (200). Additionally, we verify if the data returns to the user through a file creation process involving the personal data, presuming the file's machine-readability. The developed query, checking for all derived code properties (Section 3.3) is shown in Listing 1.3 and a plot of the data flows described above is illustrated in Figure 3.

```
MATCH path1=(psi:PseudoIdentifier)--()-[:DFG*]->(hr1:HttpRequest {name:
    "POST"})-[:TO]->(he1:HttpEndpoint)-[:DFG*]->(d1:DatabaseQuery {type
    :"CREATE"})
WHERE NOT EXISTS
 {MATCH path2=(psi)--()-[:DFG*]->(hr2:HttpRequest {name: "GET"})-[:TO]->(
     he2:HttpEndpoint {method: "GET"})-[:DFG*]->(d2:DatabaseQuery {type:"
     READ"})-[:DFG*]->({name: "HttpStatus.OK"}),
     path3=(:FileWrite)-[:CALLS]->(m:MemberCallExpression)-[:ARGUMENTS
         ]->(:Node)<-[:DFG*]-(hr2)
    WHERE (d1)-[:STORAGE]->(:DatabaseStorage)<-[:STORAGE]-(d2)}
RETURN path1
```

Listing 1.3. Cypher query that detects Article 20(1) non-compliant personal data flows. `path1` maps tagged personal data stored in a database. `path2` traces these data flows to a query retrieving the data from the same database. `path3` finds member call expressions using this data to create a files on the disk. Non-compliance returns `path1` if `path2` or `path3` are missing.

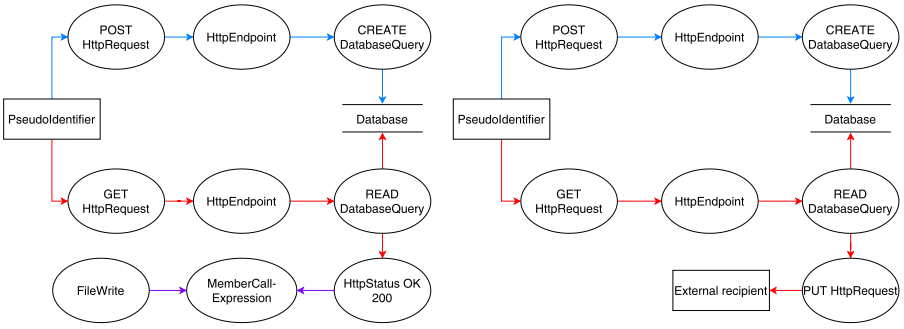


Fig. 3. Diagram consisting of two data flow diagrams – Left data flow diagram: Shows the three data flows `path1` (blue), `path2` (red) and `path3` (purple) of the query for the non-compliance check of Article 20(1). Right data flow diagram: Illustrating the two data flows `path1` (blue) and `path2` (red) of the query for the non-compliance check of Article 20(2).

The second paragraph specifies that the user can also arrange that the personal data is transmitted directly from the data controller itself to another data controller in a machine-readable format. We check for every personal data that is stored in the database if there exists a data flow that leads an HTTP request of type GET. This HTTP request then leads to an HTTP endpoint, which subsequently leads to a database query that loads the personal data from the same database. Furthermore, we investigate whether the personal data loaded from the database is now passed to an HttpRequest of type PUT, which does not have an HTTP endpoint and thus communicates to an external recipient. The query, checking for all derived code properties (Section 3.3) is shown in Listing 1.4 and the data flow is illustrated in Figure 3.

```
MATCH path1=(psi:PseudoIdentifier)--()-[:DFG*]->(hr1:HttpRequest {name:
    "POST"})-[:TO]->(he1:HttpEndpoint)--()-[:DFG*]->(d1:DatabaseQuery {
    type: "CREATE"})
WHERE NOT EXISTS
 {MATCH path2=(psi)--()-[:DFG*]->(hr2:HttpRequest {name: "GET"})-[:TO]->(
    he2:HttpEndpoint)--()-[:DFG*]->(d2:DatabaseQuery {type:"READ"})-[:
    DFG*]->(hr3:HttpRequest {name: "PUT"})
    WHERE NOT (hr3)-[:TO]-(:HttpEndpoint) AND (d1)-[:STORAGE]->(:
        DatabaseStorage)<-[:STORAGE]-(d2)}
RETURN path1
```

Listing 1.4. Cypher query that identifies Article 20(2) non-compliant data flows. `path1` tracks tagged personal data in a database. `path2` traces data flows from `path1` to a query that loads and sends this data via HTTP to an external party. `path1` indicates non-compliance if `path2` is absent.

5 Evaluation and Discussion

5.1 Accuracy

For assessing the accuracy of detecting (non-)compliance using our queries, we developed a dedicated test suite, as no such test suite existed for GDPR-specific code samples. It comprises 19 test cases split into two categories, i.e., for testing compliance and for testing non-compliance. The tests are based on scenarios and code properties from Section 3.

Each test case in our suite is structured as a Python program, consisting of a client and a server, performing communication with each other. Additionally, each case includes a configuration file with simulated deployment data, like mock databases. The PPG is then used to generate a code property graph from the Python code and storing the results in a Neo4j database. The Cypher queries presented in Section 4.2 are then executed and compared against expected results. For instance, the Article 16 test case models a client-server interaction handling personal data, with a focus on data rectification, and aims to ensure no data flow contravenes Article 16 of the GDPR.

Our findings, summarized in Table 1, indicate the test suite's effectiveness in detecting non-compliances, albeit with some false positives and a false negative. The causes of these inaccuracies and potential improvements are discussed in Section 5.4. This test suite offers a foundation for future research to refine static analysis tools and discuss the implications of the GDPR on code level.

Art.16	**Art.17(1)**	**Art.17(2)**	**Art.19**	**Art.20(1)**	**Art.20(2)**
■ □	■ □	■ □	■ □	■ □ □	■ □

Table 1. Results of the evaluation of the compliance checks using the implemented test suite: ■ = Expected result, successful detection of (non-)compliance; □ = False positive; □ = False negative

5.2 Performance

As the evaluation of the original PPG shows [12], its performance primarily depends on used passes instead of memory, retrieval of results or the pure parsing of the code. Therefore, we evaluate the impact of newly written or modified passes on the execution time of the PPG. For this purpose, the software state of the PPG before the extension of the passes is first applied to the Python code of one test case of Article 20(1). This test case contains HTTP requests, database operations and a write file call and thus triggers the DatabaseOperationPass, HttpRequestPass and FileWritePass, which were extended in the scope of this paper. Finally, the PPG and the extensions are applied to the same code part and the results are compared. The benchmark involves 20 iterations, including one

warm-up iteration not counted in the final measurement. We used a MacBook Pro with Intel Core i7 processor of 2018 with 16gb of RAM for the evaluation.

Before extending passes, execution times varied from 685ms to 875ms, averaging 742ms (SD = 28ms). After extensions, times ranged from 777ms to 879ms, with an average of 839ms (SD = 26ms), marking a 97ms increase. Despite the rise, we would argue that this difference is negligible, since the execution of the tool is generally not time-critical.

5.3 Reduction of Effort through Automation

Compliance verification usually involves manual review of code and documentation, a process that often is labor-intensive and error-prone. By automating compliance checks, a more consistent and in-depth analysis is possible that allows legal experts to make better informed decisions and allows to retest applications quickly for potential non-compliances.

For programmers, the usage of automated compliance checks reduces the effort required by developers in two key ways. Due to the automated compliance checks, programmers receive immediate feedback on GDPR compliance of their code. This immediate response makes it easier for developers to address compliance issues as they code, rather than having to revisit large sections of the codebase for compliance reviews at a later stage. Automated compliance checks significantly streamline the interaction between legal professionals and programmers by reducing the need for extensive explanations, how a certain article can be reflected into source code, which is also necessary after changes made to the software. Typically, translating complex legal requirements of GDPR into a language that is understandable to programmers can be a challenging and time-consuming process. With automated checks, this translation is inherent in the PPG and highlights affected data flows.

In order to use the automated compliance checks, it is first necessary to setup the PPG, create new passes, if certain programming languages or libraries are not yet supported and to train developers to correctly insert `PseudoIdentifier` labels in code. We do not discuss the effort related to these steps, since these are introduced by the usage of the PPG itself and not from the compliance checks. A discussion of these aspects can be found in the paper of Kunz et al. [12].

5.4 Limitations

In this paper, we have derived code properties based on an example scenario. While the example is designed in a generic way to achieve broad applicability, there is a possibility that the example might not encompass the full landscape of real-world application variations. E.g., our current implementation focuses on communication over HTTP, but also other communication protocols are used in practice, like HTTPS, FTP, Telnet or SMTP. To enable the detection of these protocols as well, additional passes for each respective protocol need to be added to the PPG and the queries must be adjusted.

Another limitation is that the proposed workflows represent only one approach to achieve GDPR compliance for the constructed running example. Often, multiple options exist for complying with GDPR articles, such as Article 17's right to data deletion: Our workflow assumes user-initiated deletion via an UI interaction, but alternative compliant methods, like email requests, exist. The construction of the running example and the derivation of code properties from proposed workflows thus can lead to false negatives.

Another limitation of our approach is that while it ensures the elements required for GDPR compliance are present at the process level, it does not guarantee the correctness of their implementation. For example, in the case of the right to rectification (Art. 16), our approach can verify that an data flow to an update operation exists, but it cannot ensure that the operation correctly updates the intended rows in the database, potentially leading to incorrect or incomplete updates.

Another limitation to note is that our implementation heavily relies on the implementation of the *CPG* and the PPG. This reliance inherently means that the accuracy and effectiveness of our work depends on the correct implementation of these tools. More significantly, any inherent limitations or shortcomings present within the *CPG* and PPG approaches could be replicated in our own implementation and can lead to false results in non-compliance detection. An example, which leads to false positive results, arises from how the PPG handles annotations in the source code. It creates a separate graph node for each *PseudoIdentifier* annotation. Consequently, identical personal data processed in different files results in multiple nodes, potentially missing necessary data flows for compliance. For example let us consider a web app with distinct user registration and data editing pages, both handling the same personal data, but recognized as separate by the PPG. This could falsely indicate non-compliance (false positive) in some scenarios. A proposed solution is to assign a unique ID to each *PseudoIdentifier* annotation, allowing the PPG to merge identical data annotations into a single node, reducing false positives.

Another limitation involves the query for Article 20(1) non-compliance detection. The current focus is on storage of personal data in a file, presuming a machine-readable format (code property **Prop-json-data-format**). However, without validating this property, the query might overlook scenarios where data is not stored in a JSON format, leading to false negatives. Addressing this, we suggest developing a new pass that abstracts file descriptor opening, verifying if a file is in a machine-readable format.

Also annotating personal data in the code is crucial for successful compliance checks. This allows the PPG to store and check associated nodes. However, human errors in labeling can lead to false positives and negatives. Training for developers on what should be considered "personal data" is thus essential.

Furthermore, the accuracy of compliance check results, as evaluated by our self-implemented test suite, may have inherent biases due to the limited variety of code examples tested. We thus plan to enrich the test suite with diverse test cases, addressing various compliance scenarios (see Section 6).

6 Conclusion

In this paper, we introduced an innovative method for automated GDPR compliance checks using a code property graph. First, we have translated GDPR requirements into code properties that can be automatically detected. We then extended an existing static code analysis tool [12] to incorporate these properties, enabling automated verification. Our tool can thus integrate into automated software development workflows and assist legal experts in compliance assessment. Testing with a 19-case test suite confirmed the tool's effectiveness and practicality in identifying compliant and non-compliant code segments.

In future work we will refine the tool by expanding the test suite for more complex cases, improving reliability in detecting GDPR issues, and minimizing errors. Finally, we want to extend our tool to cover more GDPR articles.

Acknowledgments. This work was partly funded by the German Federal Ministry for Economic Affairs and Climate Action, within the project ToHyVe.

References

1. GDPR checklist for data controllers. `https://gdpr.eu/checklist/`. Accessed: 2024-03-25.
2. Ala'a Al-Momani, Kim Wuyts, Laurens Sion, Frank Kargl, Wouter Joosen, Benjamin Erb, and Christoph Bösch. Land of the lost: privacy patterns' forgotten properties: enhancing selection-support for privacy patterns. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1217–1225, 2021.
3. Steven Arzt, Stephan Huber, Siegfried Rasthofer, and Eric Bodden. Denial-of-app attack: Inhibiting the installation of android apps on stock phones. *Proceedings of the ACM Conference on Computer and Communications Security*, 2014.
4. Christian Banse, Immanuel Kunz, Angelika Schneider, and Konrad Weiss. Cloud Property Graph: Connecting Cloud Security Assessments with Static Code Analysis. *IEEE International Conference on Cloud Computing, CLOUD*, 2021-Septe:13–19, 2021.
5. Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. A privacy threat analysis framework: Supporting the elicitation and fulfillment of privacy requirements. *Requirements Engineering*, 16(1):3–32, 2011.
6. European Parliament and European Council. General Data Protection Regulation: GDPR. In *Official Journal of the European Union*, number 119, pages 1–88, 2016.
7. Pietro Ferrara, Luca Olivieri, and Fausto Spoto. *Tailoring Taint Analysis to GDPR*. Springer International Publishing, Cham, 2018.
8. Rafa Galvez and Seda Gurses. The odyssey: Modeling privacy threats in a brave new world. In *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 87–94, 2018.
9. Kalle Hjerppe, Jukka Ruohonen, and Ville Leppänen. *Annotation-Based Static Analysis for Personal Data Protection*. Springer International Publishing, 2020.
10. Jaap-Henk Hoepman. Privacy design strategies. In Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans, editors, *ICT Systems Security and Privacy Protection*, pages 446–459. Springer Berlin Heidelberg, 2014.

11. Edward Kost. 10-step checklist: Gdpr compliance guide. https://www.upguard.com/blog/how-to-be-gdpr-compliant. Accessed: 2024-03-25.
12. Immanuel Kunz, Konrad Weiss, Angelika Schneider, and Christian Banse. Privacy Property Graph: Towards Automated Privacy Threat Modeling via Static Graph-based Analysis. *Proceedings on Privacy Enhancing Technologies*, pages 171–187, 2023.
13. Prashant Mahajan. Launching privado open source for privacy compliance and data security. https://privado.ai/post/launching-privado-open-source-for-privacy-compliance-and-data-security. Accessed: 2023-04-19.
14. Feiyang Tang and Bjarte M. Østvold. *Assessing Software Privacy Using the Privacy Flow-Graph*. MSR4P&S 2022. Association for Computing Machinery, New York, NY, USA, 2022.
15. Kim Wuyts, Laurens Sion, and Wouter Joosen. Linddun go: A lightweight approach to privacy threat modeling. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 302–309. IEEE, 2020.
16. Fabian Yamaguchi, Nico Golde, Daniel Arp, and Konrad Rieck. Modeling and discovering vulnerabilities with code property graphs. *Proceedings - IEEE Symposium on Security and Privacy*, pages 590–604, 2014.

Privacy-preserving tabular data generation: Systematic Literature Review

Pablo Sanchez-Serrano, Ruben Rios, and Isaac Agudo

Network, Information and Computer Security (NICS) Lab, University of Malaga, Spain {pablosanserr,ruben.rdp,isaac}@uma.es

Abstract. There is a wide range of tabular data of great value to science, economy and social progress. When sharing such data, privacy must be taken into account. Traditionally, this has been addressed through anonymization. However, in recent years, with the growth of AI, the possibility of using generative models has emerged as a way to generate synthetic data that guarantees privacy while maintaining their utility. This systematic literature review aims to identify and classify existing privacy-preserving tabular generative models in order to create a taxonomy of solutions. In addition, we analyze the privacy metrics and techniques they use, and identify possible unexplored lines of research.

Keywords: Synthetic data · Privacy · Tabular data.

1 Introduction

There is a wide variety of tabular data, including medical records, financial transactions, and demographic details. This data holds immense value for scientific, economic and social progress, as it can be used to identify patterns, facilitate decision-making and disseminate knowledge. However, the sharing of this data raises privacy concerns, given that it often contains PII (personally identifiable information).

Traditional methods for protecting privacy in tabular data include [20]: data pseudonymization, which replaces PII with fake identifiers, and data anonymization, which involves generalization, suppression and perturbation techniques that modify attributes in the dataset to obtain a supposedly anonymous dataset. To decrease the risk of re-identification some models like k-anonymity, l-diversity and t-closeness have been proposed. Recently, generative models have emerged as a way to guarantee the privacy of datasets [9]. These models generate synthetic data from real datasets, mimicking the statistical properties of the training data.

When dealing with synthetic datasets, there are significant differences in the amount of knowledge and access available to different users (see Fig. 1). This involves a range of privacy challenges that need to be considered. Users further to the right of the diagram show a higher level of difficulty in discerning which data were used to generate the synthetic data. The number of barriers will be higher the further to the right the user is located, i.e. the less knowledge and access the user has.

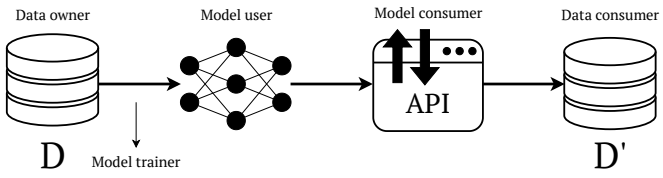


Fig. 1. Different levels of knowledge and access to the trained model.

A *model trainer* uses the real data (D) given by the data owner to train a generative model. The model trainer must be careful with possible data leakage due to errors or intermediate outputs. The model trainer could also be malicious, or the data owners may not trust the data owner. Security mechanisms such as homomorphic encryption [3] or federated learning [36] should be implemented. Once the model is trained, the user can have different levels of access to the model. We refer to the user with full access to the model as the *model user*. Despite having completed the training phase, it may be possible to obtain information about D from the model [38]. Conversely, a *model consumer* can only generate samples from the model using an API, but do not have access to the trained model. The amount of information available to this type of users depends on the API. A first-level API allows unlimited samples generation, leading to honest-but-curious users who seeks information while respecting established protocols. On the other hand, a second-level API has some restrictions on data generation, i.e. limited number of requests or attributes that are not allowed to be generated. Membership Inference Attacks (MIAs) [26] can exploit the lack of restrictions on data generation. MIAs take advantage of differences in how models respond to queries from members inside and outside of the training dataset. Finally, the *data consumer* only has access to a synthetic dataset (D') generated by the model, and is unable to generate samples by himself. Although more challenging, it is possible to obtain information about D from D' [4].

The contributions of this paper can be summarized as follows:

1. The use of a systematic methodology to provide an overview of privacy techniques used in tabular data generative models.
2. A collection of 24 systematically selected papers.
3. A collection of privacy metrics for in tabular data generative models.
4. A taxonomy of privacy-preserving generative models for tabular data.

This works is organized as follows. Section 2 introduces the methodology and how the papers were selected. Section 3 discusses the different ways to measure privacy in tabular data generation and explains the techniques used to ensure privacy collected from the selected papers. Section 4 provides a taxonomy of generative models for tabular data, giving an order and clarifying the differences between them. Finally, Section 5 draws conclusions and outlines possible lines of future research based on the observations made in the paper.

2 Systematic literature review

A Systematic Literature Review (SLR) is a rigorous approach to reviewing and synthesizing research literature on a specific topic. This methodology is designed to provide a comprehensive, unbiased and reproducible summary of existing research. The PICOC framework is employed to define the scope and focus of our study. It involves three main steps: planning, conducting and reporting.

2.1 Planning

This SLR is performed to answer the following questions:

1. What are the main techniques used to guarantee privacy in generative models for tabular data?
2. How can we measure the privacy of generative models for tabular data?

PICOC terms help to define a list of keywords, as shown in Table 1. Using these keywords we can create a search query (see Definition 1), which addresses our research questions.

Table 1. Keyword list created from PICOC terms.

Keywords	Synonyms	PICOC
Tabular data	Database, Dataset	Population
Privacy techniques	Data masking, Differential privacy, Masked data, Privacy approach, Privacy methods, Privacy-preserving, k-anonymity, l-diversity, t-closeness	Intervention
Generative model	Data synthesis, Synthesizer, Synthetic data generation, Synthetic generator	Comparison
Benchmark		Outcome
Privacy metric	Anonymity metric	Outcome
Utility metric	Data quality, Data utility, ML efficacy, Usefulness of data	Outcome

Definition 1 (Search Query). *("Tabular data" OR "Database" OR "Dataset") AND ("Privacy techniques" OR "Data masking" OR "Differential privacy" OR "Masked data" OR "Privacy approach" OR "Privacy methods" OR "Privacy-preserving" OR "k-anonymity" OR "l-diversity" OR "t-closeness") AND ("Generative model" OR "Data synthesis" OR "Synthesizer" OR "Synthetic data generation" OR "Synthetic generator") AND ("Benchmark" OR "Privacy metric" OR "Anonymity metric" OR "Utility metric" OR "Data quality" OR "Data utility" OR "ML efficacy" OR "Usefulness of data")*

The next step is to define which digital libraries use to search. We selected IEEE Digital Library, ISI Web of Science and Scopus. There might be duplicate papers but this will be taken into account in the conducting phase.

To refine the search and ensure the inclusion of high-quality and relevant studies, the following exclusion criteria are applied: (i) accepted papers should address privacy for generative AI models for tabular data, (ii) surveys or reviews will be discarded, (iii) only articles, conference papers, proceedings or journals will be considered, (iv) a minimum number of citations is required. Papers published before 2022 should have at least 20 citations. Papers from 2022 are required to include a minimum of 10 citations. Papers from 2023 or 2024 must have a minimum of 5 citations. To sum up, these are the exclusion criteria:

- The paper does not discuss privacy
- The paper does not discuss AI
- The paper does not focus on tabular data
- It is a survey/review
- It is not an article, conference paper, proceeding or journal
- It has not enough citations
- It is not published in English

After an initial filtering using the exclusion criteria, a checklist of five questions (listed below) with specific criteria is established. There are three possible scores for each criterion: Yes (1 point), Partially (0.5 points), or No. Thus, 5 points is the maximum score. Papers that reach 3 points are finally selected.

1. Does the article propose a new AI model for tabular data generation?
2. Does the article propose new attacks to privacy in generative models?
3. Does the paper propose a model practical implementation?
4. Does the model include techniques to provide privacy?
5. Does the article discuss how to measure privacy for tabular generative data models? Does it also include a way to measure utility?

2.2 Conducting

The first step is to perform a search using the query string presented in Section 2.1. Initially, a total of 977 papers were found. From this list of papers, 36 were duplicated, giving a total of 941 unique papers. To provide a clearer understanding of the evolution of research on this topic, Figure 2 illustrates the number of papers published each year. The graph shows a growth in the number of papers over the years. Although the number of papers published in 2024 is lower than in previous years, the reason is that the current writing date is mid 2024.

This is the moment to apply the exclusion criteria presented in Section 2.1. All papers are reviewed, focusing on the title, keywords, and abstract. At the end of this process, 61 papers are accepted.

After an initial filtering, it is time to apply the Quality Assessment Checklist presented in Section 2.1. During this step, potential papers are added through snowballing. The papers added in this way are also submitted to Quality Assessment Checklist. During the conducting process, a backward snowballing (or backward reference searching) is performed. This involves looking through the references listed in the selected papers to find older studies that the key papers

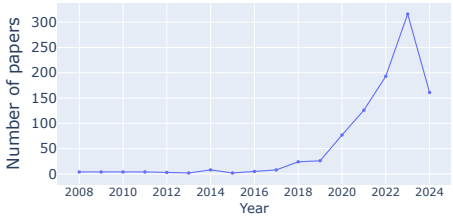


Fig. 2. Number of papers found

Table 2. Reference list of papers.

Years	Papers
2017	[15]
2019	[2], [34], [11]
2020	[33], [12]
2021	[6], [27], [13], [5]
2022	[28], [31], [32], [30], [8], [16]
2023	[22], [14], [35] [17], [29], [18], [19]
2024	[37]

have cited, which might also be relevant in the research topic. At the end of this process, a final list of 24 papers are selected. The reference list of papers is shown in Table 2. As with the papers found with the query (Figure 2) there is an increase in the number of selected papers over the years, except in 2024.

2.3 Reporting

In this section, we extract some statistical data about the selected articles. The information extracted from the papers is discussed in the following sections.

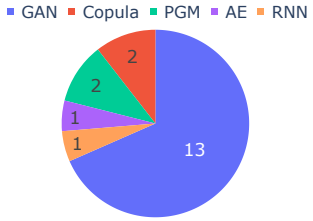


Fig. 3. Model family distribution, in which models are grouped according to their nature or type.

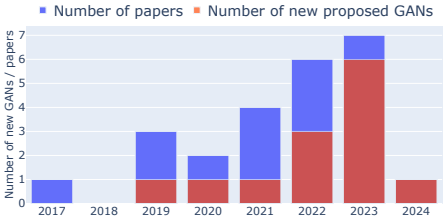


Fig. 4. Evolution of the GANs proposed in the selected papers compared to the number of selected papers.

Out of the 24 selected papers, 17 papers propose a new model for privacy-preserving tabular data generation. There are two papers that propose two models, for a total of 19 proposed models. Figure 3 shows the different types of model families collected. This chart will be useful in establishing a taxonomy of different generative models. There is a clear predominance of GANs over the others.

Figure 4 compares the years of creation of GANs with the years of publication of all selected papers. It can be seen that the growth of interest in GANs follows the growth of interest in the research area. This shows that GANs are the type of generative models that are most often used to generate tabular data with privacy guarantees.

3 Measuring Privacy in tabular data generation

There are several ways to measure privacy in generative models for tabular data. Some traditional privacy techniques, such as k-anonymity or t-closeness, can also implicitly act as privacy measures. Among the selected papers, differential privacy stands out.

Differential privacy [7] is a mathematical framework designed to provide privacy guarantees for data entries within a dataset. Differential privacy ensures that the inclusion or exclusion of a single individual's data does not significantly affect the outcome of any analysis, thereby protecting the individual's privacy.

Definition 2 (Neighboring Datasets). *Two datasets, D and D', are neighboring, if and only if D' differs from D in only one entry.*

Definition 3 ((ε, δ)-Differential Privacy). *For a non-negative privacy budget ε and a non-negative relaxation term δ, an algorithm, M, satisfies (ε, δ)-differential privacy if for any pair of neighboring datasets D, D' and $S \subseteq Range(M)$*

$$Pr[M(D) \in S] \leq exp(\varepsilon) \cdot Pr[M(D') \in S] + \delta \tag{1}$$

where Pr is taken with respect to the randomness of M. δ is a relaxation term to ε-differential privacy. There are a variety of techniques for achieving differential privacy. Essentially, the algorithm M perturbs the input with some noise distribution, i.e. normal distribution, based on ε and δ.

The following expression is obtained by clearing ε from expression 1:

$$\varepsilon \geq ln\left(\frac{Pr[M(D) \in S] - \delta}{Pr[M(D') \in S]}\right) \tag{2}$$

A lower value of ε implies a higher level of privacy because inequality 2 is more restrictive. However, decreasing ε increases the noise that needs to be added to satisfy Definition 3

There are some variations or extensions of the definition of differential privacy, such as RDP (Rényi Differential Privacy) [21], LDP (Local Differential Privacy) or CDP (Concentrated Differential Privacy).

Privacy accounting concept indicates that there is a need of some "accountant" procedure that computes the privacy cost at each access to the training data, and accumulates this cost as the training progress [1]. The privacy analysis of our some differential privacy techniques employs the moments accountant approach to keep track of the privacy cost in multiple iterations. This concept can also be used to measure privacy degradation with increasing number of queries. One way to compensate for this progressive loss of privacy would be to progressively increase the noise.

There are several techniques to ensure differential privacy, such as Differentially Private Expectation Maximization (DP-EM) [25], Private Aggregation of Teacher Ensembles (PATE) [23,24] or Differentially Private Stochastic Gradient

Descent (DP-SGD) [1]. In general, they all involve the addition of noise in one way or another.

Similar to differential privacy, there is also the concept of identifiability [33]. This framework is used to measure and limit the risk of re-identification. There are also other ways to measure privacy for those models that do not theoretically guarantee privacy, but rather focus on an empirical approach to measure privacy. These focus on performing attacks to see how effective they are. The most common is the Membership Inference Attack (MIA) [26].

SELENA [30] is a ensemble method that combines Split-AI and Self-Distillation to mitigate MIAs. Although SELENA is primarily designed for supervised classification tasks, it could be used as a component of a generative model. For example, SELENA could be used in GANs to protect the discriminator from revealing membership information about the training data. SELENA trains submodels on random data subsets and uses adaptive inference to ensure similar behavior on member and non-member inputs, significantly reducing MIA risks.

4 A taxonomy for tabular data generative models

This section categorizes tabular data generative models from selected papers (see Figure 5). Due to length restrictions, the taxonomy focuses on GANs with privacy guarantees. However, other types of models were found:

- Autoencoders (AEs): DP-SYN [2]
- Probabilistic Graphical Models (PGMs): PrivMRF [5] and PrivIncr [18]
- Recurrent Neural Networks (RNNs): Conditional-LSTM [22]
- Copula-based models: LoCop and DR_LoCop [32]

The white boxes in Figure 5 represent each of the 13 models, while the gray boxes represent the categories into which the different models fall. Note that DP-GAN, whose connector is shown as a dotted line, is a particular case. Although it is possible to introduce conditions on one of its components [13], it does not fall within the definition of a conditional GAN. Therefore, it is placed in the category of non-conditional GANs. Models that were originally designed to generate EHR (Electronic Health Record) data are in a green box. Similarly, those GANs that integrate an autoencoder as a component of their model are in a blue box.

4.1 Generative adversarial networks (GAN)

A generative adversarial network (GAN) [10] is a type of machine learning framework where two neural networks are trained simultaneously in a zero-sum game setting. GANs have established themselves as one of the state-of-the-art generative models. GANs consists of two adversarial models:

- Generator G: takes random noise as input and generates samples. It aims to generate data that imitates a given dataset.

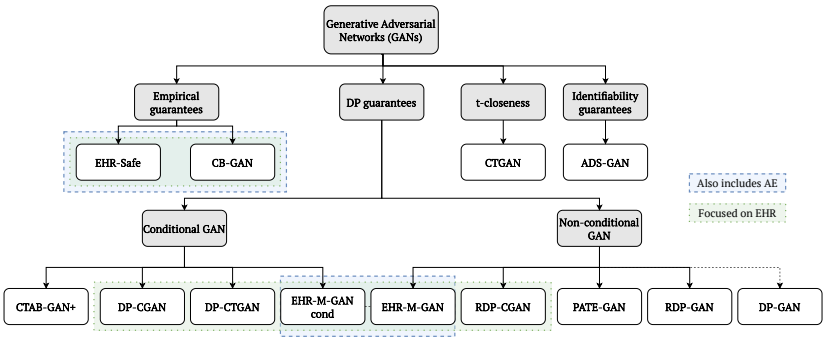


Fig. 5. Privacy-preserving tabular data GAN taxonomy

– Discriminator D: attempts to differentiate between real data samples taken from the training dataset and fake data samples generated by the generator. It outputs a probability indicating if a given sample is real or fake.

The generator tries to fool the discriminator by generating realistic data. The discriminator tries to become better at distinguishing real data from fake data. This creates a minimax game between them. The generator aims to maximize the probability of the discriminator misclassifying its outputs as real, and the discriminator aims to minimize the probability of incorrectly classifying real data as fake and vice versa.

There is a wide variety of GANs, each one specialized in generating certain kinds of data, such as images, video, network trafic, tabular data, etc.

Conditional GANs There is no control on the process of data generation in a standard GAN. It generates synthetic data from the real data without allowing any further conditions or requirements. Conditional Generative Adversarial Networks (CGANs) are used to address this problem. With CGANs, a condition can be included to control the data generation process. The following types of CGANs are designed to generate tabular data ensuring differential privacy:

– CTAB-GAN+ [37]: It is a general purpose model trained with DP-SGD to impose strict privacy guarantees and leverage the RDP for privacy accounting because it provides stricter bounds on the privacy budget.
– DP-CGAN [29]: It is focused on EHR data generation. This model uses standard differential privacy.
– DP-CTGAN [8]: It is focused on EHR data generation. This model also uses standard differential privacy. Has a federated learning-oriented variant, FDP-CTGAN.
– EHR-M-GAN cond [17]: It is a conditional variation of EHR-M-GAN. It is focused con EHR data generation. It uses a dual variational autoencoder (dual-VAE) as a part of its architecture. DP-SGD is used to guarantee privacy.

Non-conditional GANs There are other ways to create synthetic data with privacy assurances beyond CGANs. The following GAN models provide privacy guarantees but are not conditional:

- EHR-M-GAN [17]: It is focused con EHR data generation. It uses a dual variational autoencoder (dual-VAE) as a part of its architecture. It uses DP-SGD to guarantee privacy.
- DP-GAN [13]: One of the components is a conditional network, but it is not a conditional GAN as CGANs are defined. This model uses standard differential privacy.
- PATE-GAN [34]: This model modify the discriminator to be differentially private using a modified version of PATE framework.
- RDP-CGAN [31]: It is a convolutional GAN focused on EHR data. To ensure privacy, this model uses RDP.
- RDP-GAN [19]: This model uses RDP to ensure privacy. It is a general purpose model.

5 Conclusions

This paper provides an overview of the state of the art in privacy-preserving tabular data generation. From a total of 941 unique papers, we selected 24 papers to answer two research questions: "What are the main techniques used to guarantee privacy in generative models for tabular data?" and "How can we measure the privacy of generative models for tabular data?". For the first question, we found that although there is a wide range of generative models in the literature, GAN is the predominant model for synthetic tabular data generation, and the most used application scenario is the protection of medical records. Regarding the second question, most models focus on providing differential privacy guarantees, either its standard definition or some variants. However, we also found some models that do not theoretically guarantee privacy, but rather focus on an empirical approach to measure privacy. As future work, we plan to identify other generative models where the community has not yet begun to discuss privacy risks, and analyze the reasons for this, in order to incorporate privacy guarantees into these models.

Acknowledgments. This work has been partially supported by project PID2022-139268OB-I00, financed by MCIN/AEI /10.13039/501100011033 / FEDER, UE and project TED2021-129830B-I00, financed by MCIN/AEI /10.13039/501100011033/Next-GenerationEU/PRTR.

References

1. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep Learning with Differential Privacy. In: Conference on Computer and Communications Security (CCS). p. 308–318. ACM (2016)

2. Abay, N.C., Zhou, Y., Kantarcioglu, M., Thuraisingham, B., Sweeney, L.: Privacy Preserving Synthetic Data Release Using Deep Learning. In: Machine Learning and Knowledge Discovery in Databases. pp. 510–526. Springer (2019)
3. Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jäschke, A., Reuter, C.A., Strand, M.: A Guide to Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2015/1192 (2015)
4. van Breugel, B., Sun, H., Qian, Z., van der Schaar, M.: Membership inference attacks against synthetic data through overfitting detection (2023), `https://arxiv.org/abs/2302.12580`
5. Cai, K., Lei, X., Wei, J., Xiao, X.: Data synthesis via differentially private markov random fields. Proc. VLDB Endow. **14**(11), 2190–2202 (2021)
6. Domingo-Ferrer, J., Muralidhar, K., Bras-Amorós, M.: General Confidentiality and Utility Metrics for Privacy-Preserving Data Publishing Based on the Permutation Model. IEEE Trans on Dependable and Secure Computing **18**(5), 2506–2517 (2021)
7. Dwork, C.: Differential privacy. In: Automata, Languages and Programming. pp. 1–12. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
8. Fang, M.L., Dhami, D.S., Kersting, K.: DP-CTGAN: Differentially Private Medical Data Generation Using CTGANs. In: Artificial Intelligence in Medicine. pp. 178–188. Springer (2022)
9. Figueira, A., Vaz, B.: Survey on Synthetic Data Generation, Evaluation Methods and GANs. Mathematics **10**(15) (2022)
10. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial networks. Commun. ACM **63**(11), 139–144 (2020)
11. Hittmeir, M., Ekelhart, A., Mayer, R.: Utility and Privacy Assessments of Synthetic Data for Regression Tasks. In: IEEE Conference on Big Data. pp. 5763–5772 (2019)
12. Hittmeir, M., Mayer, R., Ekelhart, A.: A Baseline for Attribute Disclosure Risk in Synthetic Data. In: ACM Conference on Data and Application Security and Privacy (CODASPY). p. 133–143. ACM (2020)
13. Ho, S., Qu, Y., Gu, B., Gao, L., Li, J., Xiang, Y.: DP-GAN: Differentially private consecutive data publishing using generative adversarial nets. Journal of Network and Computer Applications **185**, 103066 (2021)
14. Hu, R., Li, D., Ng, S.K., Zheng, Z.: CB-GAN: Generate Sensitive data with a Convolutional Bidirectional Generative Adversarial Networks. In: Database Systems for Advanced Applications. pp. 159–174. Springer Nature (2023)
15. Jia, R., Sangogboye, F.C., Hong, T., Spanos, C., Kjærgaard, M.B.: PAD: protecting anonymity in publishing building related datasets. In: ACM Conference on Systems for Energy-Efficient Built Environments (BuildSys). ACM (2017)
16. Kotal, A., Piplai, A., Chukkapalli, S.S.L., Joshi, A.: PriveTAB: Secure and Privacy-Preserving sharing of Tabular Data. In: International Workshop on Security and Privacy Analytics (IWSPA). p. 35–45. ACM (2022)
17. Li, J., Cairns, B.J., Li, J., Zhu, T.: Generating synthetic mixed-type longitudinal electronic health records for artificial intelligent applications. NPJ Digital Medicine **6**(1), 98 (2023)
18. Liu, G., Tang, P., Hu, C., Jin, C., Guo, S., Stoyanovich, J., Teubner, J., Mamoulis, N., Pitoura, E., Mühlig, J.: Multi-dimensional data publishing with local differential privacy. In: EDBT. pp. 183–194 (2023)
19. Ma, C., Li, J., Ding, M., Liu, B., Wei, K., Weng, J., Poor, H.V.: RDP-GAN: A Rényi-Differential Privacy Based Generative Adversarial Network. IEEE Trans on Dependable and Secure Computing **20**(6), 4838–4852 (2023)

20. Majeed, A., Lee, S.: Anonymization Techniques for Privacy Preserving Data Publishing: A Comprehensive Survey. IEEE Access **9**, 8512–8545 (2021)
21. Mironov, I.: Rényi Differential Privacy. In: IEEE Computer Security Foundations Symposium (CSF). pp. 263–275 (2017)
22. Mosquera, L., El Emam, K., Ding, L., , et al.: A method for generating synthetic longitudinal health data. BMC Medical Research Methodology **23**(1), 67 (2023)
23. Papernot, N., Abadi, M., Úlfar Erlingsson, Goodfellow, I., et al.: Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data (2017)
24. Papernot, N., Song, S., Mironov, I., Raghunathan, A., Talwar, K., Úlfar Erlingsson: Scalable Private Learning with PATE. In: Int. Conf. on Learning Representations (ICLR) (2018)
25. Park, M., Foulds, J., Choudhary, K., Welling, M.: DP-EM: Differentially Private Expectation Maximization. In: International Conference on Artificial Intelligence and Statistics. vol. 54, pp. 896–904. PMLR (2017)
26. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: Symposium on Security and Privacy (IEEE S&P). pp. 3–18 (2017)
27. Song, L., Mittal, P.: Systematic Evaluation of Privacy Risks of Machine Learning Models. In: 30th USENIX Security Symposium. pp. 2615–2632 (2021)
28. Stadler, T., Oprisanu, B., Troncoso, C.: Synthetic Data – Anonymisation Groundhog Day. In: 31st USENIX Security Symposium. pp. 1451–1468. Boston, MA (2022)
29. Sun, C., van Soest, J., Dumontier, M.: Generating synthetic personal health data using conditional generative adversarial networks combining with differential privacy. Journal of Biomedical Informatics **143**, 104404 (2023)
30. Tang, X., Mahloujifar, S., Song, L., Shejwalkar, V., Nasr, M., et al.: Mitigating Membership Inference Attacks by Self-Distillation Through a Novel Ensemble Architecture. In: 31st USENIX Security Symposium. pp. 1433–1450 (2022)
31. Torfi, A., Fox, E.A., Reddy, C.K.: Differentially private synthetic medical data generation using convolutional GANs. Information Sciences **586**, 485–500 (2022)
32. Wang, T., Yang, X., Ren, X., Yu, W., Yang, S.: Locally Private High-Dimensional Crowdsourced Data Release Based on Copula Functions. IEEE Trans on Services Computing **15**(2), 778–792 (2022)
33. Yoon, J., Drumright, L.N., van der Schaar, M.: Anonymization Through Data Synthesis Using Generative Adversarial Networks (ADS-GAN). IEEE Journal of Biomedical and Health Informatics **24**(8), 2378–2388 (2020)
34. Yoon, J., Jordon, J., van der Schaar, M.: PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In: Int. Conf. on Learning Representations (ICLR) (2019)
35. Yoon, J., Mizrahi, M., Ghalaty, N.F., et al.: EHR-Safe: generating high-fidelity and privacy-preserving synthetic electronic health records. NPJ Digital Medicine **6**(1), 141 (2023)
36. Zhang, C., Xie, Y., Bai, H., Yu, B., Li, W., Gao, Y.: A survey on federated learning. Knowledge-Based Systems **216**, 106775 (2021)
37. Zhao, Z., Kunar, A., Birke, R., Van der Scheer, H., Chen, L.Y.: CTAB-GAN+: enhancing tabular data synthesis. Frontiers in Big Data **6** (2024)
38. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)

A DPIA Repository for Interdisciplinary Data Protection Research

Laurens Sion[0000−0002−8126−4491], Dimitri Van Landuyt[0000−0001−6597−2271], and Wouter Joosen[0000−0002−7710−5092]

DistriNet, KU Leuven, 3001 Leuven, Belgium
firstname.lastname@kuleuven.be

Abstract. Any data collection or processing activity that incurs significant risk requires a Data Protection Impact Assessment (DPIA), which is a comprehensive, analytical evaluation of the risks of violating fundamental data protection rights. While performing a DPIA is considered a cornerstone activity for demonstrating GDPR compliance and adherence to data protection by design principles, they are rarely made public by organizations. Although DPIAs have received considerable attention from a wide range of inter-disciplinary research perspectives, this attention remains fragmented and a solid comparative basis does not yet exist.
In this paper, we present our efforts in establishing an open repository that indexes common and representative DPIAs. Starting from clearly-defined inclusion requirements for representative cases, we present the outcome of the first two years of consolidation efforts: a repository indexing 130 DPIAs. Finally, we discuss how this repository enables interdisciplinary DPIA research, on comparing and evaluating diverse DPIA approaches, models, and tools. The resulting repository is a valuable resource for researchers across disciplines, to spark debate on DPIA goals and quality, and to evaluate different DPIA methodologies, approaches, and tools.

Keywords: data protection · DPIA · privacy · repository.

1 Introduction

With its introduction in 2018, the GDPR has re-affirmed the importance of addressing privacy and data protection concerns early in the design and development of software-intensive systems that collect or process personal data. One of its obligations in cases with high risk –but a good practice regardless– is to perform a Data Protection Impact Assessment (DPIA). This activity starts with systematically describing the data processing operations and then involves an in-depth assessment of the risks they pose to data subjects' rights and freedoms.

While many national supervisory authorities provide guidance, advice, and templates for DPIA reports; these resources remain more generic in nature and do not provide normative illustrations of the potential or desired results of such an assessment in practice. Hence, it is hard for organizations to gain an appreciation of the expected outcomes of a concrete DPIA in terms of detail and depth.

Many scientific communities have made efforts to center their collective focus on exemplars [33,23], common and public data sets, problem characterizations [24], and reference benchmarks [17,40,35]. The establishment and consolidation of common, accessible resources is essential to create a community effect, to foster discussion, and to enable empirical research efforts.

Similar efforts however are severely lacking in the research on DPIAs. In recent years, many tools [8,34], guidelines [16,4,9,10,11], templates [3], commercial offerings [27,5,32,1,28] and academic studies have emerged. In the work-up towards the GDPR, the Article 29 Working Party [4] has established a number of criteria and guidelines for DPIAs, but these are still relatively open-ended. While all of these elements add value towards providing guidance, a common baseline or shared understanding in terms of, for example, the degree of completeness or level of detail is lacking. This problem is exacerbated by the fact that organizations rarely publish outcomes: DPIAs are a risk analysis activity that may convey sensitive details about the organizations.

To address these limitations, we present the approach and results of establishing a public repository of DPIAs, currently consisting of 130 DPIAs.[1] We discuss the goals, inclusion criteria, and desired attributes of representative cases.

2 Related work

Performing a DPIA is necessary in data processing operations that are "*likely to result in a high risk to the rights and freedoms of natural persons*" (Art. 35(1)) and is a generally accepted technique for a controller to meet its obligations to appropriately manage risk [4], regardless of necessity. DPIAs can be used to assess a single processing operation, multiple similar or related ones, or to assess the data protection impact of a specific technology or product. A wide range of approaches, tools, methodologies, and recommendations [25] have been discussed.

Domain-specific approaches and cases. A number of approaches have been proposed to conduct a DPIA in specific application domains. One area of active focus is the performance of DPIA in health-related systems. Georgiou et al. [19] discuss the application of the CNIL methodology for a DPIA of cloud-based health applications. Várkonyi and Gradišek [37] use a DPIA to assess the risks inherent to the use of Artificial Intelligence (AI) in an e-health context. Conte et al. [13] present a case study called Health360 involving the application of a DPIA to Electronic Health Records. Alnemr et al. [2] have presented a dedicated approach and tool (DPIAT) for performing DPIAs in cloud applications. Bisztray et al. [6] present an in-depth analysis of the applicability of two DPIA approaches to biometric-based authentication systems. Other applications of DPIAs focus on cyber-physical systems [22], big data applications [20], the use of AI [37] or biometrics [6], smart cities [7], smart grids [30], but also vertical application domains and sectors such as that of charities [21]. Vandercruysse et al. [36]

[1] This repository is available at `https://dpiarepository.distrinet-research.be`.

discuss the merits of applying DPIAs to particular enabling technologies and even hardware-based systems.

Methodologies, approaches and tools. The performance of DPIA is traditionally decoupled from the software development life-cycle, performed by a more compliance-oriented stakeholder such as a data protection officer (DPO). This decoupling is generally undesirable as contemporary development (agile development, CI/CD) is highly incremental which leads to the DPIA itself becoming outdated very quickly. To avoid such a divergence, a number of tools and techniques have recently emerged. Among these are model-based approaches that involve the creation of intermediate artifacts [26,8,34,12,14], but also machine-readable representations such as the Data Privacy Vocabulary (DPV) [29]. Some tools and approaches [15] even focus on integrating these activities with the source code.

Field studies. Wright et al. [39] have performed an in-depth comparison of the adoption and implementation of Privacy Impact Assessment (PIA) activities in six different countries, focusing on both differences and commonalities. Their study highlights the value of a PIA as an activity, and emphasizes that these activities are ideally not performed one-off, but part of a more continuous process. The PIA report itself in that sense as an artifact is just one element of the broader context of a PIA, and the process (e.g., accountability, internal review, etc.) is considered equally important. The descriptive field study of van Puijenbroek and Hoepman [31] focuses on practitioners in the Netherlands. It illustrates the diversity of application domains, in approaches and methodologies (specifying operations and assessing risk) and highlights the current lack of harmonization and more concrete reproducible guidelines. Friedewald et al. [18] share experiences gained from several specific and practical DPIA efforts and highlight the value of more interdisciplinary effort (e.g., through more extensive integration of DPIA outcomes in technology-centric risk assessment approaches and tools).

3 Requirements for a DPIA repository

We first list and articulate the main requirements for the open shared DPIA repository presented in this article. Section 4 further motivates these requirements on a per-stakeholder basis.

1 *Quality assessment and control.* The DPIA reports selected for the repository should include mature DPIA examples of high quality.
2 *Case inclusion and coverage.*
  2.a *Representativeness.* The repository should include representative examples of typical or common application cases, and these should not be artificial.
  2.b *Diversity.* The repository should contain a wide range of DPIAs for a variety of data processing operation types.
3 *Methodological Coverage.*
  3.a *Representativeness.* The repository should include DPIAs resulting from sufficiently representative methodologies that are used in practice.

3.b *Diversity.* The repository should have examples of DPIA outcomes of different methodologies and approaches.

4 *Alternative DPIA artifact for a single case.* In addition to accepting intermediate artifacts contributing to the establishment of a single DPIA entry, the repository should provide support for accepting different or alternative DPIA outcomes (models, spreadsheets, etc.) for the same case.

5 *Multiple DPIA artifact versions over time.* The repository should support keeping track of different versions of the DPIA artifacts over time.

4 Motivating use cases

An open, shared, and accessible repository of DPIA outcomes supports and strengthens interdisciplinary scientific research, but has purposes beyond purely research. This section discusses the motivating use cases[2] for different types of stakeholders.

Practitioners currently lack access to rich and normative DPIA examples. Even for examples that are available, information is lacking on the quality and this prohibits them from selecting cases that can be considered normative and positive. Practitioners require access to enriched examples, to allow them to better grasp what is expected of them, in terms of comprehensiveness, detail, and argumentation (req 1). In addition, an open, diverse, and representative collection of DPIAs allows practitioners to search for the most relevant examples based on similarity to their applications (req 2.a).

DPIA Researchers can study the diversity in approaches on how to structure and motivate different data processing operations (req 3.b) and explore and evaluate scientific hypotheses (for example, completeness in terms of covering the WP29 criteria [4]). Furthermore, it allows the development of quality criteria of DPIAs (e.g., comprehensiveness and depth of the analysis) (req 3.b). Alternative artifacts (req 4) for a single case also enable comparative evaluation of the quality of the DPIA outcomes, while, inclusion of multiple versions (req 5) enables longitudinal analysis. Once there is consensus about the quality of individual artifacts, it paves the way for nominating exemplary and normative reference cases.

Software vendors and service providers can access DPIA outcomes to help determine what information to provide to their customers for DPIA efforts and to assist them in performing DPIAs involving their products or services (reqs 1 and 2.a).

Data subjects will benefit from a public DPIA repository to gain insight and appreciation for the measures taken by organizations that process their personal data and, ideally, to demand similar efforts from other organizations that process their data (reqs 1 and 2.a). For concerned data subjects, including a DPIA in a public repository, and opening it up to public scrutiny can increase trust in the organization's intentions towards respecting key data subjects' rights and freedoms.

[2] References to numbered requirements from Section 3 are included between brackets.

5 A community repository of DPIAs

This section elaborates on the repository of DPIAs in four parts: the implementation, the DPIA meta-data, and the current set, and operating the repository.

5.1 Design and implementation of the repository

This section elaborates on the development of the repository itself to host those DPIAs and their metadata. The main criteria are the following: (i) prefer static sites to avoid complex hosting requirements; (ii) provide support for processing/hosting collections of items; (iii) enable the custom specification of metadata or properties; (iv) input data should be structured in an accessible format (e.g., CSV, YAML, etc.); and (v) be relatively easy to customize the output.

The framework we encountered that best meets these criteria is *Collection-Builder* [38], an open source framework for creating digital collections that leverages the Jekyll static site generator. This makes it especially to create the DPIA index in a publicly-available repository on, for example, GitHub or GitLab.

Only a few implementation steps are needed to customize the output of the collection items to provide all the additional attributes (Section 5.2) on the generated DPIA entry pages. The largest effort involves populating and enriching the collection items with their metadata.

5.2 DPIA attributes

We enrich the collected DPIAs with a rich set of meta-data attributes. This allows users to perform search queries and supports advanced navigation of the repository. The attributes are grouped into the following categories (Tables 1 and 2): (i) the described processing operations, (ii) the involved organizations, (iii) the DPIA report or artifacts, and (iv) coverage of the WP29 DPIA criteria [4].

5.3 Current set of DPIAs

At the initial release of the DPIA repository in 2022, it consisted of 25 entries. This amount has grown throughout 2023 to 41 and currently contains 130 DPIAs.

The initial set of DPIAs was gathered by conducting search queries for '`data protection impact assessment`' and '`data protection impact assessment report`' and through further snowballing (by including related DPIAs when they were encountered). These queries were constrained to PDF, further discarding templates and guidelines in the search results. These results are then further complemented with DPIAs encountered by the authors. This initial set is to be expanded as part of a continuing community effort over time.

The collection is not yet representative for the full range of published DPIAs, or the diversity of methods and approaches that we aim at covering. In addition, the complete determination of the different attributes outlined in Section 5.2 is a still a work in progress as this will –amongst other efforts– require the peer review

Table 1. Overview of DPIA attributes

	I. Intrinsic complexity of the processing operation	
1	**Indirect collection**	The amount of indirect collections of personal data.
2	**Disclosure**	The amount of disclosures of personal data to other parties.
3	**Automated decision-making**	The amount of automated decision-making activities that occur.
4	**Sensitive personal data**	The amount of sensitive types of personal data are processed as part of the data processing operations.
5	**Sensitive personal data types**	The concrete types of sensitive personal data that are processed.
6	**Data subjects**	The amount of data subject types considered and a characterisation (e.g., adults and minors).
7	**Types of data subjects**	The concrete types of data subjects involved in the data processing operations.
8	**Cross-border transfers**	The amount of cross-border data transfers involved in the data processing operations.
9	**Processing size**	The size of the described processing operations in terms of the amount of different processing operations.
	II. Involved organizations	
1	**Controller**	The amount of controllers.
2	**Processor**	The amount of (sub-)processors.
3	**Controller countries**	The countries where the controllers are established.
4	**Processor countries**	The countries where the (sub-)processors are established.
5	**Organization types**	The types of involved organizations (e.g., government, company)
6	**Joint controllership**	Whether multiple controllers are involved in the processing operation as joint controllers.
7	**Representatives**	The amount of organizations involved in the processing that need to have representatives in the EU.
	III. Regarding the DPIA artifact	
1	**Year**	The year of the DPIA. This is the year when finished, not when the assessment started.
2	**Language**	The language in which the DPIA is available.
3	**Report size**	The size of the PDF report in number of pages.
4	**Scope**	The scope: a single processing operation, multiple similar processing operations, or a technology product?
5	**Template**	The particular template used to create the DPIA.
6	**Method**	The methodology used to perform the DPIA.
7	**Tool execution**	The application or framework used to perform the DPIA.
8	**Performed by**	Who performed the DPIA.
9	**Artifact type**	Type of artifact (e.g., PDF, spreadsheet, model).
10	**Version**	The version of the DPIA.
11	**Processing ID**	An ID for the processing that can be used to link multiple repository entries to the same case.

Table 2. Overview of DPIA attributes (continued)

	IV. WP29 DPIA CRITERIA	
1	**Processing description**	Contains a systematic description of the processing. This can be further broken down into: (i) the nature, scope, context, and purposes; (ii) the personal data, recipients, and period of storing; (iii) the processing operations; (iv) the assets (hardware, software, people, etc.); and (v) compliance with codes of conduct.
2	**Necessity and proportionality (processing)**	The DPIA describes measures contributing to the proportionality and the necessity of the processing. This can be further broken down into: (i) specified, explicit and legitimate purposes; (ii) the lawfulness of the processing. (iii) what is necessary data. and (iv) storage limitations.
3	**Necessity and proportionality (data subject rights)**	The DPIA describes measures contributing to the rights of data subjects. This can be further broken down into: (i) information provided to the data subject; (ii) right of access and data portability; (iii) rectification and erasure; (iv) objection and restriction of processing; (v) relationships with processors; (vi) safeguards surrounding international transfer(s); and (vii) prior consultation.
4	**Risk to the rights and freedoms of data subjects**	The DPIA describes the risks (origin, nature, particularity, and severity). This can be further broken down into: (i) the risk sources; (ii) the potential impacts to data subjects' rights and freedoms; and (iii) the estimated likelihood and severity of those risks.
5	**Risk to the rights and freedoms of data subjects (measures)**	The report describes specific measures to treat those risks (mitigate, reduce, manage).
6	**Interested parties**	The DPIA describes the involved interested parties. This can be further broken down into: (i) the DPO advice; and (ii) views of data subjects or their representatives.

for quality control (outlined in Section 5.4). Nonetheless, we argue that this is a significant and relevant stepping stone towards fostering the open collaboration and the community effect that is required for this endeavor. In the next section, we outline our vision on how the repository can be operated in support of this.

5.4 Operating the DPIA repository

This section briefly outlines the operation of the repository. Although the current version of the repository is not yet the result of a peer review process, we have anticipated and designed an explicit process for community interaction and peer review of the DPIAs.

Table 3. Qualitative assessment of requirement coverage

Requirement (Attributes)	Rationale
1 Quality (IV.1–6)	The quality of the selected DPIAs in the repository is mainly ensured through the process for including new DPIAs. Quality assessment and peer review ensures that new submissions to the repository are of sufficient quality and are paired with rich meta-data in terms of the attributes from Table 1. The WP29 DPIA criteria are a good set of independent properties to assess the quality of the DPIA report in terms their coverage (as described in the DPIA).
2.a Case representativeness (I.1–9, II.1–7)	The representativeness of the cases described in the DPIAs submitted to the repository can be assessed using the DPIA attributes of categories I and II. These are assessed as part of the submission process to ensure the described processing operations are representative.
2.b Case diversity (I.1–9, II.1–7)	The diversity of cases in the repository cannot be assessed for an individual DPIA, but needs to be continually assessed in terms of the attributes of categories I and II to ensure that the set of DPIAs in the repository contains enough variation over these attributes.
3.a Method representativeness (III.4–8)	The method representativeness can mainly be assessed through the artifact properties, to the extent that they document or describe the process that was followed. The procedure for including new DPIA can ensure that the followed methodologies are relevant and representative.
3.b Method diversity (III.4–9)	The diversity in methods is assessed over the repository in terms of the attributes in category III.
4 Artifact types (III.5–7, III.9–11)	The repository supports capturing multiple different artifact types as separate entries and link them to the same processing operations (III.11)
5 Artifact versioning (III.10–11)	The version information and link to the same processing operation that is being described allows the repository to capture longitudinal information of how a processing operation is described in different DPIAs over time.

Submission. Upon submission, the properties, involved artifacts, diversity and representativeness (reqs 2.b and 2.a) are verified.

Review. The second phase entails the community review of the attributes by other legal stakeholders to ensure agreement and correctness. This review enables more extensive discussions in terms of the quality and the identification of exemplary DPIAs in terms of, for example, the comprehensiveness and depth of their system description or legal rationale.

Publication & Maintenance. After inclusion and publication, repository management will be a continuous effort to maintain data quality.

6 Conclusion

In this paper, we highlight the current lack of high-quality, normative examples and rich, realistic application cases and discuss how it impedes scientific research on DPIAs. We particularly motivate the value of a public and accessible collection of DPIA artifacts for a variety of stakeholders, ranging from practitioners and technology providers, to tool developers, to DPIA researchers, and to data subjects.

As the main contribution, we present the design and implementation of a DPIA repository, open for collaboration and community contribution and report its current state. Having been maintained for two years, the repository currently consists of 130 DPIAs. Table 3 discusses how the requirements (Section 3) can be attained through the repository (Section 5) and its operation.

This paper is a first necessary stepping stone in a longer-term community-building effort on maintaining and enriching the overall repository of DPIAs. We strongly believe that interdisciplinary collaboration will be required to further grow this collection not just in size, but also in terms of documented attributes, intermediate and alternative representations, etc. As a secondary effect of this effort, we argue that a public, comparative repository may also nudge organizations into further increasing their DPIA efforts, to publish high-quality DPIAs, thereby further increasing the transparency of their data processing operations and reducing overall data protections risks. Finally, to further illustrate the scientific value of the repository, we highlight that we are currently conducting an in-depth comparative evaluation of a model-driven DPIA framework called DPMF [34] using DPIA artifacts selected entirely from the repository.

Acknowledgments. This research is partially funded by the Research Fund KU Leuven and Cybersecurity Research Program Flanders.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Akarion AG: Niobase: Master the GDPR. `https://niobase.com/en/` (12 2019)
2. Alnemr, R., Cayirci, E., Corte, L.D., Garaga, A., Leenes, R., Mhungu, R., Pearson, S., Reed, C., Oliveira, A.S.d., Stefanatou, D., et al.: A data protection impact assessment methodology for cloud. In: Annual Privacy Forum. Springer (2015)
3. Article 29 Working Party: Opinion 07/2013 on the data protection impact assessment template for smart grid and smart metering systems (2013)
4. Article 29 Working Party: Guidelines on data protection impact assessment (DPIA) (WP248 rev.01) (2017)
5. AvePoint: AvePoint Privacy Impact Assessment System: Comply with GDPR and other key data protection regulations (12 2019), `https://www.avepoint.com/privacy-impact-assessment/`
6. Bisztray, T., Gruschka, N., Mavroeidis, V., Fritsch, L.: Data protection impact assessment in identity control management with a focus on biometrics. In: Open Identity Summit 2020. Lecture Notes in Informatics, Gesellschaft für Informatik e.V. (2020)

7. Calvi, A.: Gender, data protection & the smart city: Exploring the role of dpia in achieving equality goals **19**(3), 24–47 (2022)
8. CNIL: PIA: Analyse d'impact sur la protection des données (Privacy Impact Assessment). Commission Nationale de l'Informatique et des Libertés (2018), https://www.cnil.fr/en/open-source-pia-software-helps-carry-out-data-protection-impact-assesment
9. CNIL: Privacy Impact Assessment (PIA) 1 : Methodology. Commission Nationale de l'Informatique et des Libertés (2 2018), https://www.cnil.fr/sites/default/files/atoms/files/cnil-pia-1-en-methodology.pdf
10. CNIL: Privacy Impact Assessment (PIA) 2 : Template. Commission Nationale de l'Informatique et des Libertés (2 2018), https://www.cnil.fr/sites/default/files/atoms/files/cnil-pia-2-en-templates.pdf
11. CNIL: Privacy Impact Assessment (PIA) 3 : Knowledge Bases. Commission Nationale de l'Informatique et des Libertés (2 2018), https://www.cnil.fr/sites/default/files/atoms/files/cnil-pia-3-en-knowledgebases.pdf
12. Coles, J., Faily, S., Ki-Aries, D.: Tool-supporting data protection impact assessments with cairis. In: 2018 IEEE 5th International Workshop on Evolving Security & Privacy Requirements Engineering (ESPRE). pp. 21–27. IEEE (2018)
13. Conte, R., Sansone, F., Tonacci, A., Pala, A.P.: Privacy-by-design and minimization within a small electronic health record: The health360 case study. Applied Sciences **12**(17) (2022), https://www.mdpi.com/2076-3417/12/17/8441
14. Dashti, S., Ranise, S.: Tool-assisted risk analysis for data protection impact assessment. In: IFIP International Summer School on Privacy and Identity Management. pp. 308–324. Springer (2019)
15. Ethyca: Fides: The open-source language for data privacy. https://ethyca.com/fides
16. European Data Protection Supervisor: Accountability on the ground part II: Data Protection Impact Assessments & Prior Consultation (2018)
17. Ficco, M., Rak, M., Venticinque, S., Tasquier, L., Aversano, G.: Cloud evaluation: Benchmarking and monitoring. Quantitative Assessments of Distributed Systems pp. 175–199 (2015)
18. Friedewald, M., Schiering, I., Martin, N., Hallinan, D.: Data protection impact assessments in practice. In: European Symposium on Research in Computer Security. pp. 424–443. Springer (2021)
19. Georgiou, D., Lambrinoudakis, C.: Data Protection Impact Assessment (DPIA) for Cloud-Based Health Organizations. Future Internet **13**(3) (2021), https://www.mdpi.com/1999-5903/13/3/66
20. Gruschka, N., Mavroeidis, V., Vishi, K., Jensen, M.: Privacy Issues and Data Protection in Big Data: A Case Study Analysis under GDPR. In: 2018 IEEE International Conference on Big Data (Big Data). pp. 5027–5033 (2018)
21. Henriksen-Bulmer, J., Faily, S., Jeary, S.: Implementing GDPR in the Charity Sector: A Case Study, pp. 173–188. Springer International Publishing, Cham (2019)
22. Henriksen-Bulmer, J., Faily, S., Jeary, S.: DPIA in context: applying DPIA to assess privacy risks of cyber physical systems. Future internet **12**(5), 93 (2020)
23. Iftikhar, M.U., Ramachandran, G.S., Bollansée, P., Weyns, D., Hughes, D.: Deltaiot: A self-adaptive internet of things exemplar. In: 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) (2017)
24. Katz, S., Mezini, M.: Transactions on Aspect-Oriented Software Development VII: A Common Case Study for Aspect-Oriented Modeling, vol. 6210. Springer (2010)

25. Kloza, D., Van Dijk, N., Gellert, R.M., Borocz, I.M., Tanas, A., Mantovani, E., Quinn, P.: Data protection impact assessments in the european union: complementing the new legal framework towards a more robust protection of individuals (2017)
26. Meis, R.: Problem-based privacy analysis (ProPAn) – a computer-aided privacy requirements engineering method. Ph.D. thesis (12 2018)
27. Nymity: Automate the entire PIA process and provide better response time to the businesswith improved business engagement. `https://www.nymity.com/wp-content/uploads/Nymity-PIA-DPIA-Datasheet.pdf` (12 2019)
28. OneTrust: OneTrust Privacy Management Software. `https://www.onetrust.com/solutions/privacy-compliance/` (12 2019)
29. Pandit, H.J., Esteves, B., Krog, G.P., Ryan, P., Golpayegani, D., Flake, J.: Data privacy vocabulary (dpv) – version 2 (2024)
30. Piatkowska, E., Bajraktari, A., Chhajed, D., Smith, P.: Tool support for data protection impact assessment in the smart grid **134**(1), 26–29 (2017)
31. van Puijenbroek, J., Hoepman, J.H.: Privacy impact assessments in practice: Outcome of a descriptive field research in the netherlands (2017)
32. RealDPG: RealDPG features and benefits. `https://www.realdpg.com/en/features-benefits` (12 2019)
33. Shin, Y.J., Liu, L., Hyun, S., Bae, D.H.: Platooning legos: An open physical exemplar for engineering self-adaptive cyber-physical systems-of-systems. In: 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). pp. 231–237. IEEE (2021)
34. Sion, L., Dewitte, P., Van Landuyt, D., Wuyts, K., Valcke, P., Joosen, W.: DPMF: a modeling framework for data protection by design. Enterprise Modelling and Information Systems Architectures (EMISAJ) **15**, 10–1 (2020)
35. Van Landuyt, D., Levrau, M., Reniers, V., Joosen, W.: An e-commerce benchmark for evaluating performance trade-offs in document stores. In: Proceedings of the 26th International Conference on Big Data Analytics and Knowledge Discovery
36. Vandercruysse, L., Buts, C., Dooms, M.: Beyond Data Controllership: Merits of a Generic DPIA by Hardware and Technology Suppliers. Eur. Data Prot. L. Rev. **6**, 133 (2020)
37. Várkonyi, G.G., Gradišek, A.: Data protection impact assessment case study for a research project using artificial intelligence on patient data. Informatica **44**(4) (2020)
38. Williamson, E.P., Becker, D., Wikle, O.: CollectionBuilder-CSV (4 2021), `https://github.com/CollectionBuilder/collectionbuilder-csv`
39. Wright, D., Finn, R., Rodrigues, R.: A comparative analysis of privacy impact assessment in six countries **9**(1) (2013)
40. Zhang, C., Lu, J., Xu, P., Chen, Y.: Unibench: A benchmark for multi-model database management systems. In: Technology conference on performance evaluation and benchmarking. pp. 7–23. Springer (2018)

Secrecy and Sensitivity: Privacy-Performance Trade-Offs in Encrypted Traffic Classification

Spencer Giddens*[1†], Raphael Labaca-Castro*[2‡], Dan Zhao[3†], Sandra Guasch[2], Parth Mishra[2], and Nicolas Gama[2]

[1] University of Notre Dame, IN, USA
sgiddens@nd.edu, *joint first author
[2] SandboxAQ, Palo Alto, CA, USA
raphael.labaca@sandboxaq.com, *joint first author
{sandra.guasch, parth.mishra, nicolas.gama}@sandboxaq.com
[3] New York University, NY, USA
dz1158@nyu.edu

Abstract. As datasets and models grow in size and complexity to increase performance, the risks associated with sensitive data also grow. Differential privacy (DP) offers a framework for designing mechanisms that provide a degree of privacy that can help conceal sensitive features or information. However, different domains and applications can naturally exhibit different rates of trade-offs between privacy and performance depending on their characteristics. In contrast to well-studied areas (e.g., healthcare), one relatively unexplored domain is network traffic analysis where the data contains sensitive information on users' communications. In this paper, we apply DP to various machine learning models trained to classify between encrypted and non-encrypted packets from network traffic; we emphasize that our goal is to examine a relatively unexplored area to analyze the trade-offs between privacy and performance when the data contains both encrypted and un-encrypted observations. We show how varying model architecture and feature sets can be a relatively simple way to achieve more optimal performance-privacy trade-offs; we also compare and contextualize reasonable privacy budgets from our analysis in the network traffic domain against those in other more well-studied domains.

Keywords: network traffic classification · differential privacy · privacy budget · performance evaluation.

1 Introduction

Network traffic analysis is a key component of infrastructure security—proper identification of network protocols can facilitate network sizing and enable the detection of anomalous connections, revealing ongoing attacks or insecure protocols within the network.

† This publication is a result of the work done during their stay at SandboxAQ.
‡ Corresponding author.

A unique challenge for training machine learning (ML) models on network traffic data lies with the data itself, which contains sensitive information such as IPs, ports, protocols, or clear-text payloads. If these models are shared across different parties, it is imperative that no sensitive information on the underlying data is leaked.

In this paper, we explore the trade-offs of applying differential privacy (DP) [9] to a ML model to protect the privacy of the data used to train it, in the context of network traffic classification into plain and encrypted traffic. The main reason behind it is to be able to detect the use of unsecure protocols and unencrypted communications within an infrastructure, which may have security and legal / compliance consequences. Although the use of zero-trust architectures and encrypted communications by default is increasing, there may be environments where these are still not enforced due to the increased management and configuration complexity, and the potential penalisation in performance.

By varying the choice of model architecture and features used, we study the privacy-performance trade-offs of training both the DP and non-DP versions of models and show how these changes, along with the underlying domain and data characteristics, can considerably impact the selection of reasonable privacy budgets.

2 Background

2.1 Network Traffic

Data is carried over computer networks in the form of discrete network packets where each packet carries protocol-dependent headers along with information in its payload. These packets then constitute a tuple-like structure consisting of multiple fields of information including source and destination IP addresses, ports, and other data relevant to network protocols. These packets are our fundamental unit of information as we classify between encrypted and un-encrypted/plain traffic.

Although deterministic solutions already exist for distinguishing between plain and encrypted traffic based on the protocol, entropy calculation, ports, and other features, these solutions do not work well in corner cases especially when compressed data is evaluated. Compressed values present higher entropy and therefore are often indistinguishable from encrypted data. Any solution relying on only such a metric would be prone to higher false positive values. This is our main motivation for using machine learning for this classification.

We note that if a compressed file is sent over an un-encrypted protocol we consider it plain.

2.2 Differential Privacy (DP)

Differential privacy, intuitively, ensures that for any given individual in a dataset, the output of a DP-satisfying mechanism will be similar whether the individual's

(a) An example of a plain network packet: a data unit transmitted over a computer network without any additional headers, encapsulation, or encryption.

(b) An example of an encrypted network packet: a packet that has been encoded or scrambled to protect content from unauthorized access or interception.

Fig. 1: An illustration of the two types of packets: encrypted and un-encrypted (or plain).

data is included in the mechanism input or not. Those protected by DP guarantees (i.e., the entity whose presence is to be concealed) are known as privacy units and can represent any entity in the data (e.g., a user). In this paper, our privacy unit is a network packet or the information in said packet. We first formalize the notion of DP.

Definition 1 ((ε, δ)-differential privacy) [8] *A randomized mechanism $\mathcal{M}$ satisfies* (ε, δ)-DP *if for all $S \subset \text{Range}(\mathcal{M})$ and all neighboring datasets D and $\tilde{D}$ (datasets differing by a single individual),*

$$P(\mathcal{M}(D) \in S) \leq e^{\varepsilon} P(\mathcal{M}(\tilde{D}) \in S) + \delta, \tag{1}$$

where $\varepsilon > 0$ and $\delta \in [0, 1)$ are privacy budget parameters. When $\delta = 0$, we denote this as ε-DP*.*

The degree of similarity between neighboring datasets for a DP mechanism is governed by ε; smaller values correspond to more privacy and vice versa. The parameter δ is commonly viewed as the probability with which ε-DP fails and is usually set on the order of $o(1/\text{poly}(n))$, where n is the size of the dataset. DP's popularity can be largely attributed to its strong theoretical properties, relative ease of use, and overall flexibility. We refer to [10] for a more comprehensive review.

2.3 Related Work

Ad hoc anonymization methods alone have been insufficient to ensure privacy for sensitive datasets [15,1]. Sweeney [21] linked public voter records to anonymized health records from Massachusetts state employees to identify then-governor William Weld's health records. A similar attack [15] shut down the Netflix Prize competition after individuals in the anonymized competition dataset were partially de-identified. Even summary statistics of anonymized data have proven insecure as attacks on 2010 US Census statistics were able to reconstruct 46% of the records [6].

Models trained on sensitive data are also susceptible to attacks. [19] showed that, by using a black-box "target model" to synthesize training data and using

those to train "shadow models" replicating the target model, an attacker can use the shadow models and synthesized data to infer membership of a given record in the training dataset for the target model. [24] demonstrated (approximate) attribute inference is also possible using membership inference as a subroutine. If attackers have additional information (e.g., model parameters), more attacks are possible [19,24]. DP makes no assumptions on the methods used by attackers to reveal an individual's presence in a sensitive dataset. [22] proved that an attacker's membership inference (MI) advantage $\mathrm{Adv}_{\mathrm{MI}}$ (i.e., the difference between the attacker's true and false positive rates) is bounded by $\mathrm{Adv}_{\mathrm{MI}} \leq e^{\varepsilon} - 1$. Even for larger ε where theoretical MI advantage bounds no longer hold, [13] demonstrated that DP still limits state-of-the-art MI attack success rates in practice. [2] explored the use of DP to train a deep neural network to classify encrypted network traffic into classes of interest but do not attempt to classify between encrypted and plain data or study performance-privacy trade-offs using reasonable privacy budgets. [17] uses a convolutional neural network to classify traffic flows into different categories and applications. However, given that the payload is not used for the training, the authors do not consider including privacy-protection techniques such as DP to protect the training data.

2.4 DP in Network Analysis

As described in § 2.2, DP provides mathematically rigorous privacy guarantees to the data which, in this context, we use to try and protect sensitive data that is commonly exchanged throughout our networks. One problem with network data is that sensitive information might be exposed while travelling the network, for example behind TLS terminations in enterprise infrastructures. If attributes can be inferred from the data, user information such as visited websites, financial transactions, or even passwords can be revealed. Other domain-specific peculiarities include the possible existence of more efficient DP mechanisms when portions of the network data are already encrypted or applying DP to an online stream of network packet time-series data among others—all relatively unexplored areas which we leave as part of our future work.

3 Methodology

The goal is to train a binary classifier using two sets of features, with and without DP in the model being trained, to distinguish between encrypted and un-encrypted network traffic. These two sets of features can be characterized by two approaches towards where the most useful network data lies: namely *header-based*, in which information about the network packet header is extracted, and *payload-based*, that focuses in calculating metrics that characterize the network payload. We train *vanilla* versions of these models (without DP guarantees) as baselines to compare against their differentially private versions.

3.1 Approach

To classify network traffic into *encrypted* and *non-encrypted* (or plain) data, we pursue two strategies, each characterized by a different set of explanatory variables in classifying network data: a *header-based* approach and a *payload-based* approach. In the former, a number of features are extracted from the header of the packet, while, in the latter, information from the payload itself is used to calculate randomness metrics as our features.

Our privacy-preserving versions of both random forest and logistic regression models are implemented in Python via IBM's DP library, *diffprivlib* [12]. These are produced with ε-DP guarantees for various values of ε. As the dataset considered for this paper is already public, our main focus is to explore the privacy-utility trade-off to determine a reasonable domain-specific value for the privacy budget ε that balances DP guarantees and model performance. To have a fair comparison between vanilla and DP models, especially given the additional privacy budget allocation that would be necessary for hyper-parameter tuning, we train each model with its default settings.

Header-based. In this approach, the features are extracted from the header of the network packets in the dataset 3.2 including multiple fields of network protocols found in the network and transport layers. These features are calculated through a custom network dissector tool, which provides a serialized representation before entering into a pre-processing pipeline that processes the data in a way to mitigate inconsistencies produced during the data capturing process from network interfaces such as invalid or incomplete packets.

Payload-based. A payload-based approach is characterized by the hypothesis that the entropy of encrypted data will be higher than that of equal-length plain data. Inspired by [4], we use the statistics from randomness tests conducted on the payload data as features to train our classifiers. The payload is first extracted using a custom extraction algorithm before being passed into a module that conducts randomness tests on the payload including entropy, chi-squared, and arithmetic average, which are used as the key features in our payload-based approach.

Model Choices. We train random forest, decision tree, AdaBoost, and logistic regression models as our base models. Due to our specific domain and dataset, we favored tree-based approaches that tend to be well-suited with both numerical and categorical data. Likewise, we also chose to evaluate an AdaBoost algorithm since weak learners behave similar to decision trees using a single split. Due to the nature of the network data and its heterogeneity across different network environments it might be useful to leverage its iterate methods to improve overall performance. Finally, we explore the logistic regression model as a simple binary algorithm to benchmark against previous models accordingly.

3.2 Dataset

The dataset [18] consists of network data captures in PCAP format collected between July 3rd at 9AM to July 7th at 5PM in 2017 and has been used relatively frequently in the field of network traffic applications [17,20,23]. The data is labeled as encrypted or un-encrypted (plain) before being divided into train/test splits. The training set consists of ∼1.26 million packets while the test set has ∼350,000 with approximately equal representation between the encrypted/un-encrypted classes.

4 Evaluation

We evaluate the performance and trade-offs of our models with and without DP using both the feature and payload-based approaches with a 70%/30% train-test data split. Vanilla (without DP) random forests, decision trees, AdaBoost, and logistic regression models are trained; these classifiers were then evaluated via prediction accuracy and F1-score on the test set. Then, for each $\varepsilon \in \{10^{-7}, 10^{-6.5}, \ldots, 10^{3}\}$, we train 30 random forests and logistic regression models satisfying ε-DP, and calculate their accuracy and F1-scores on the test set for comparison.

Header-based approach. Figure 2 shows the results of our DP simulations. As expected, the average performance of the privacy-preserving models approaches the performance of non-DP models as ε increases. For the DP version of logistic regression, its performance approaches that of the non-DP version starting from $\varepsilon = 10^2$ onward. While theoretical privacy guarantees at these ε values are weak, [13] demonstrated that even at this large of a privacy budget, practical privacy benefits can still be realized. For the header-based approach, the random forest model appears to be more promising with only a slight performance loss. The performance metrics level off beginning around $\varepsilon = 10^{-4}$. Though at this ε we see on average a 10% performance loss, the privacy guarantees at this level are strong. Based on [22], an attacker's membership inference advantage can be at most $e^{0.0001} - 1 \approx 0.0001$, guaranteeing that an attacker's ability to infer whether any given network packet is in the training set is barely better than random guessing. Based on these results, a DP version of the random forest with the header-based approach works best.

Likely, part of the difference in the behaviour of DP random forest and DP logistic regression is due to the choice of hyperparameters such as regularization or tree depth, which poses a trade-off since hyperparameter tuning requires spending part of the privacy budget. In the case of DL logistic regression, the regularization parameter may be too large compared to the signal when computing the loss function for lower values of ϵ. In the case of DP random forest, the tree depth is limited in order to limit the privacy budget required for fitting, which puts a cap on the maximum accuracy that can be achieved for larger values of ϵ.

Payload-based approach. In Figure 2, for the payload-based approach, in contrast to the header-based comparisons, both the DP logistic regression and DP random forest reach their best performance levels at smaller values of ε. DP logistic regression begins performing similarly to its vanilla counterpart around $\varepsilon = 10^{-3}$, while the performance of DP random forest levels off at around $\varepsilon = 10^{-4.5}$. In fact, for $\varepsilon \geq 10^{-2}$, DP logistic regression even outperforms DP random forest for the payload-based approach. These values of ε all represent strong theoretical privacy guarantees against membership inference attacks.

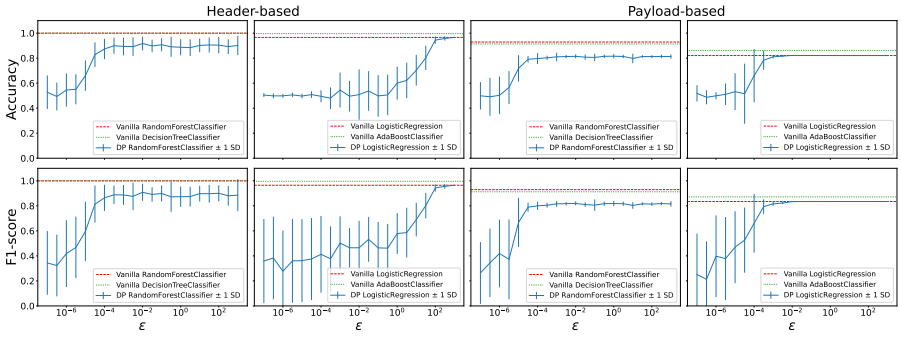


Fig. 2: Comparison of model performance between vanilla classifiers and classifiers trained with ε-DP guarantees across a range of ε privacy budget values. Both the accuracy and the F1-score are shown. The first two columns show model results from the header-based approach, while the last two pertain to the payload-based approach. The solid lines represent the average metric value over 30 seeds and the error bars represent one standard deviation in each direction. Dotted lines indicate the non-DP models' performance.

4.1 Privacy budget ε comparisons with other domains

We compare the minimum ε reasonable privacy budgets from our network traffic domain to DP models in more common, well-studied domains in finance and healthcare (Table 1).

For our purposes, we define a "reasonable privacy budget" to be a value ε at which an ε-DP classifier achieves performance that is both better than a baseline fully-random classifier and as close as possible to the performance of its analogous vanilla (non-DP) classifier. For example, a reasonable privacy budget for the DP random forest classifiers in Figure 2 would be $\varepsilon \geq 10^{-4}$. To ensure a fair comparison, we took a random sub-sample of our network traffic data of approximately the same size and class distribution and re-ran our DP model simulations. In this circumstance, the payload-based approach achieves reasonably good utility for ε values comparable to the finance domain, while the header-based

Table 1: Comparison of privacy budgets (smallest values of ε achieving acceptable utility) between our network traffic domain and other domains in other works. For a fair comparison, we adjusted the size of our training dataset to match the size of datasets from these other domains; therefore, the reasonable budgets shown for the network traffic domain here differ from those in Figure 2.

Domain	**Dataset**	**Model Type**	**Reasonable Budget**	**References**
Finance	Adult [7]	Logistic regression	$\varepsilon \geq 0.1$	[5]
Finance	Adult [7]	Decision tree	$\varepsilon \geq 1$	[11]
Healthcare	eICU [16]	Deep learning	$\varepsilon \geq 2.88$	[3]
Healthcare	Pneumonia [14]	Deep learning	$\varepsilon \geq 2.69$	[25]
Network traffic	PCAP data [18]	Logistic regression (payload)	$\varepsilon \geq 0.45$	This paper
Network traffic	PCAP data [18]	Random forest (payload)	$\varepsilon \geq 0.01$	This paper

approach struggles to obtain better-than-baseline performance for any ε in [5,11], possibly due to the dimensionality of feature sets used. Though the healthcare domain results are not directly comparable to ours due to differences in the types of differential privacy and ML models used, we believe these comparisons give additional perspective and highlight the importance of better understanding reasonable privacy budgets with respect to the particular data-generating process and characteristics of each domain.

Since our results can be used to understand which models perform better with a given privacy budget in the context of network classification, they can be used as a starting point for systems and future research dealing with classification problems with similar datasets and features, although additional fine-tuning may still be required due to the differences in specific scenarios.

5 Conclusion

In this paper, we explored the performance/privacy trade-offs in the network security domain and assessed how these trade-offs vary with the choice of features and model architecture as well as against other more well-studied domains. A better understanding of these trade-offs between performance and privacy guarantees can derive easy and efficient ways to protect sensitive data and still preserve performance. Our future work hopes to build upon this by developing more efficient privacy-preserving mechanisms such as studying DP guarantees for only protecting/concealing un-encrypted observations in datasets with both encrypted and un-encrypted data.

References

1. Ahn, S.: Whose genome is it anyway?: Re-identification and privacy protection in public and participatory genomics. The San Diego law review **52**, 751 (2015)

2. Akbari, I., Tahoun, E.: Privpkt: Privacy preserving collaborative encrypted traffic classification. ResearchGate (2020), https://doi.org/10.13140/RG.2.2.22431.59046
3. Beaulieu-Jones, B.K., Yuan, W., Finlayson, S.G., Wu, Z.S.: Privacy-preserving distributed deep learning for clinical data (2018)
4. Cha, S., Kim, H.: Detecting encrypted traffic: a machine learning approach. In: Information Security Applications: 17th International Workshop, WISA 2016, Jeju Island, Korea, August 25-27, 2016, Revised Selected Papers 17. pp. 54–65. Springer (2017)
5. Chaudhuri, K., Monteleoni, C., Sarwate, A.D.: Differentially private empirical risk minimization. Journal of Machine Learning Research **12**(29), 1069–1109 (2011), http://jmlr.org/papers/v12/chaudhuri11a.html
6. Desfontaines, D.: Demystifying the us census bureau's reconstruction attack. https://desfontain.es/privacy/us-census-reconstruction-attack.html (2021), accessed: September 5, 2023
7. Dua, D., Graff, C.: Uci machine learning repository (2017), http://archive.ics.uci.edu/ml
8. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006. pp. 486–503. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) Theory of Cryptography. pp. 265–284. Springer Berlin Heidelberg, Berlin, Heidelberg (2006), https://doi.org/10.1007/11681878-14
10. Dwork, C., Roth, A.: The algorithmic foundations of differential privacy. Foundations and Trends in Theoretical Computer Science **9**(3–4), 211–407 (aug 2014), https://doi.org/10.1561/0400000042
11. Friedman, A., Schuster, A.: Data mining with differential privacy. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 493–502. KDD '10, Association for Computing Machinery, New York, NY, USA (2010). https://doi.org/10.1145/1835804.1835868, https://doi.org/10.1145/1835804.1835868
12. Holohan, N., Braghin, S., Mac Aonghusa, P., Levacher, K.: Diffprivlib: the IBM differential privacy library. ArXiv e-prints **1907.02444 [cs.CR]** (Jul 2019)
13. Jayaraman, B., Evans, D.: Evaluating differentially private machine learning in practice. In: 28th USENIX Security Symposium (USENIX Security 19). pp. 1895–1912. USENIX Association, Santa Clara, CA (August 2019), https://www.usenix.org/conference/usenixsecurity19/presentation/jayaraman
14. Kermany, D.S., Goldbaum, M., Cai, W., Valentim, C.C.S., Liang, H., Baxter, S.L., McKeown, A., Yang, G., Wu, X., Yan, F., Dong, J., Prasadha, M.K., Pei, J., Ting, M.Y.L., Zhu, J., Li, C., Hewett, S., Dong, J., Ziyar, I., Shi, A., Zhang, R., Zheng, L., Hou, R., Shi, W., Fu, X., Duan, Y., Huu, V.A.N., Wen, C., Zhang, E.D., Zhang, C.L., Li, O., Wang, X., Singer, M.A., Sun, X., Xu, J., Tafreshi, A., Lewis, M.A., Xia, H., Zhang, K.: Identifying medical diagnoses and treatable diseases by image-based deep learning. Cell **172**(5) (Feb 2018), https://doi.org/10.1016/j.cell.2018.02.010
15. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: 2008 IEEE Symposium on Security and Privacy (sp 2008). pp. 111–125 (2008). https://doi.org/10.1109/SP.2008.33

16. Pollard, T.J., Johnson, A.E., Raffa, J.D., Celi, L.A., Mark, R.G., Badawi, O.: The eICU collaborative research database, a freely available multi-center database for critical care research. Scientific Data **5** (2018)
17. Shapira, T., Shavitt, Y.: Flowpic: Encrypted internet traffic classification is as easy as image recognition. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). pp. 680–687. IEEE (2019)
18. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp **1**, 108–116 (2018)
19. Shokri, R., Stronati, M., Song, C., Shmatikov, V.: Membership inference attacks against machine learning models. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 3–18 (2017). https://doi.org/10.1109/SP.2017.41
20. Singh, R., Srivastav, G.: Novel framework for anomaly detection using machine learning technique on cic-ids2017 dataset. In: 2021 International Conference on Technological Advancements and Innovations (ICTAI). pp. 632–636. IEEE (2021)
21. Sweeney, L.: Only you, your doctor, and many others may know. Technology Science (September 2015), `https://techscience.org/a/2015092903/`
22. Yeom, S., Giacomelli, I., Fredrikson, M., Jha, S.: Privacy risk in machine learning: Analyzing the connection to overfitting. In: 2018 IEEE 31st Computer Security Foundations Symposium (CSF). pp. 268–282 (2018). https://doi.org/10.1109/CSF.2018.00027
23. Yulianto, A., Sukarno, P., Suwastika, N.A.: Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset. In: Journal of Physics: Conference Series. vol. 1192, p. 012018. IOP Publishing (2019)
24. Zhao, B.Z.H., Agrawal, A., Coburn, C., Asghar, H.J., Bhaskar, R., Kaafar, M.A., Webb, D., Dickinson, P.: On the (in)feasibility of attribute inference attacks on machine learning models. In: 2021 IEEE European Symposium on Security and Privacy. pp. 232–251 (2021). https://doi.org/10.1109/EuroSP51992.2021.00025
25. Ziller, A., Usynin, D., Braren, R., Makowski, M., Rueckert, D., Kaissis, G.: Medical imaging deep learning with differential privacy. Scientific Reports **11**(13524) (2021), `doi.org/10.1038/s41598-021-93030-0`

Part II

CBT 2024: 8th International Workshop on Cryptocurrencies and Blockchain Technology

CBT Session 1: Layer 2 & Smart Contracts

Route Discovery in Private Payment Channel Networks

Zeta Avarikioti[1], Mahsa Bastankhah[2], Mohammad Ali Maddah-Ali[3], Krzysztof Pietrzak[4], Jakub Svoboda[4], and Michelle Yeo[5]

[1] TU Wien
georgia.avarikioti@tuwien.ac.at
[2] Princeton University, USA
mb6458@princeton.edu
[3] University of Minnesota Twin Cities
maddah@umn.edu
[4] Institute of Science and Technology Austria
{krzysztof.pietrzak, jakub.svoboda}@ist.ac.at
[5] National University of Singapore
mxyeo@nus.edu.sg

Abstract. In this work, we explore route discovery in private payment channel networks. We first determine what "ideal" privacy for a routing protocol means in this setting. We observe that protocols achieving this strong privacy definition exist by leveraging Multi-Party Computation but they are inherently inefficient as they must involve the entire network. We then present protocols with weaker privacy guarantees but much better efficiency (involving only a small fraction of the nodes). The core idea is that both sender and receiver gossip a message which propagates through the network, and the moment any node in the network receives both messages, a path is found. In our first protocol the message is always sent to all neighbouring nodes with a delay proportional to the fees of that edge. In our second protocol the message is only sent to one neighbour chosen randomly with a probability proportional to its degree. We additionally propose a more realistic notion of privacy in order to measure the privacy leakage of our protocols in practice. Our realistic notion of privacy challenges an adversary that join the network with a fixed budget to create channels to guess the sender and receiver of a transaction upon receiving messages from our protocols. Simulations of our protocols on the Lightning network topology (for random transactions and uniform fees) show that 1) forming edges with high degree nodes is a more effective attack strategy for the adversary, 2) there is a tradeoff between the number of nodes involved in our protocols (privacy) and the optimality of the discovered path, and 3) our protocols involve a very small fraction of the network on average.

Keywords: Payment Channel Networks · Privacy · Bitcoin · Route Discovery · Lightning Network

1 Introduction

Payment channel networks (PCNs) is one of the most promising approaches to scale cryptocurrencies like Bitcoin [18]. PCNs allow any pair of users to set up a payment channel with each other, thereby enabling an unlimited number of costless off-chain transactions between them. Users who are not directly connected with a payment channel can still transact with each other by routing the transaction through intermediate nodes in the network. These intermediate nodes typically charge a fee for forwarding these transactions. There are several PCN proposals [26,11,20,10,2,5]; the most widely used being the Bitcoin Lightning Network [20].

To route a transaction in a PCN from sender to receiver, a two-step process is necessary (but sometimes executed in parallel): (a) finding the optimal route or *route discovery*, which typically means finding the shortest or cheapest path from sender to receiver, and (b) executing the transaction payment in an atomic fashion, i.e., the transaction is either executed or aborted in all the channels of the path, so no party can lose money. The route discovery problem focuses on the first step, and thus the goal is to find the optimal route from sender to receiver in an efficient and privacy-preserving manner, while the atomic execution of transactions is not considered. This is a well-researched problem with several existing solutions [21,15,24,25,28,19]. However all these solutions assume that the *entire topology of the PCN is known* by at least one party (for instance the users who download the entire network [29] or trampoline nodes [21]).

Recently, however, the Bitcoin protocol upgraded to taproot [8] that uses Schnorr signatures [16] to aggregate public keys and signatures, making a transaction involving multiple users indistinguishable from a transaction involving just two users on the blockchain. As a result, the users of the Lightning Network can now form *private channels*, leading eventually to a private payment channel network with unknown topology. This in turn affects heavily the route discovery process as all existing algorithms utilize the known PCN topology. Hence, *designing route discovery algorithms suitable for private PCNs is paramount.*

Our Contribution. In this work, we consider for the first time the problem of route discovery in private PCNs where the capacities, fees, and even the mere *existence* of channels can be (partially) unknown. In particular, the route discovery protocols we propose do not assume any knowledge about the network other than the minimal requirement that nodes know about their own channels (i.e., their balances, fees, and local neighborhood).

Our objective is to construct protocols that are *efficient, private, and optimal* in terms of minimal fees. We identify several key challenges: (a) we observe that in this setting, the only strategy a sender and receiver can follow to find an (optimal) route is to send exploratory messages through the network. However, this may incur high communication overhead. (b) To find the optimal path with certainty, all nodes should be involved in the route discovery process, again leading the high communication overhead. (c) To achieve ideal privacy, nodes

that end up in a payment path should not learn any information beyond the amount and the nodes right before and after them in the path; even the sender and receiver jointly should not learn the users on a path other than their direct neighbours. Thus, no information can be leaked on the PCNs topology but even simple topology hiding broadcast protocols are highly inefficient [17,3]. (d) If ideal privacy is out of reach, there is no practical notion of privacy for PCNs to measure the possible privacy leakage.

We explain how we address these key challenges below. At its core, we exploit a trade-off between the number of involved nodes (which defines efficiency and affects privacy) and how cheap the discovered route is (optimality). To practically measure the privacy leakage, we present a novel game for route discovery protocols over arbitrary networks. In detail, our contributions are:

- (Ideal and Practical Notion) We put forward a security notion for private route discovery in Section 3 and give a feasibility result using multi-party computation (MPC). Our notion is inspired by security notions from topology hiding MPC. This solution is inefficient, not just because MPC computations are expensive, but also because it must involve the *entire* network (and this inherent for any protocol achieving our ideal notion, confirming challenge (c)). To account for the inefficiency of our ideal security notion, we also define a practical notion of privacy in Section 3.3 inspired by metrics used in information retrieval (addressing challenge (d)). We empirically analyse our protocols with respect to our practical privacy notion.
- (Practical Protocols) We present a family of route discovery protocols on private PCNs in Section 4 that are much more efficient, involving a small fraction of the network (addressing challenges (a) and (b)). These protocols work by propagating exploratory messages from the sender and receiver through the PCN. The first protocol we propose is Forward-to-All where nodes forward messages on all their edges but each edge has a delay that is proportional to its fee. In our second protocol Degree-Proportional Random Walk nodes just send messages to one neighbour, chosen randomly with a probability proportional to their degree. Forward-to-All always finds the shortest path, but it involves a larger fraction of the network than Degree-Proportional Random Walk.
- (Simulations) We simulated our protocols on the Lightning Network and a certain class of graphs (Barabási–Albert) that are used to model PCNs in Section 5. Our simulation show that Forward-to-All typically involves around 3% of nodes in Lightning, while Degree-Proportional Random Walk only involves around 0.1%, and the discovered paths are around twice as long as the optimal ones.
- (Analysis) We also prove some analytical bounds in Section 5 for our algorithms on particular classes of graphs.

In the following, we use the terms user, node, and party interchangeably.

2 Model and Definitions

We model a payment channel network (PCN) as a directed graph $G = (V, E)$ where each node in the set V represents a user in the PCN and an edge (u, v) in the set E indicates an open channel between the users u and v in V. We denote with $f_{u,v}(.)$ the fee function, i.e., u charges $f_{u,v}(x)$ to transfer x coins over the channel (u, v). In existing PCNs like the Lightning Network, $f_{u,v}(.)$ is set by u.

The route discovery problem in a PCN represents the task of finding the path with the smallest aggregated fees, or the cheapest path, in a PCN for a given pair of sender/receiver nodes $u_s, u_r \in V$ and amount x, i.e., a path $(u_0 = u_s, u_1, \ldots, u_\ell = u_r)$, minimizing the aggregated fees $\sum_{i=1}^{\ell} f_{u_{i-1},u_i}(x + \phi_{i-1})$, where ϕ_i, $i = 0, \ldots, \ell - 1$, is the aggregated fees that nodes $u_{i+1}, u_{i+2}, \ldots, u_{\ell-1}$ charge. Since the receiver u_ℓ is the last node in the path, $\phi_{\ell-1} = 0$. We use the notation $\mathsf{shortestPath}_G(u_s, u_r, x) \rightarrow \{u_0 = u_s, u_1, \ldots, u_{\ell-1}, u_\ell = u_r\}$ to describe the functionality that takes two nodes u_s and u_r and a transaction amount x as inputs and outputs the cheapest path between those two nodes.

Network model. We assume a synchronous network model, i.e., there exists some known finite time bound Δ and the adversary cannot delay delivery of any message sent by honest parties for a larger than Δ time period. We further assume all users only have local knowledge of the topology of the payment channel network with the addition of an estimate of the degree of their neighbours, i.e., each node u only knows their set of immediate neighbours and an estimate of the degree of each neighbour in that set.

3 Ideal and Practical Privacy Notions

We first define an ideal notion of privacy for route discovery and outline how to construct protocols achieving this notion, albeit very impractical ones.

Ideal privacy for PCNs means that each party only learns the bare minimum information required to participate in the transaction: its predecessor and successor on the payment path and the amount to be transferred. This information is minimal, assuming that users know at the very least the current balances on their own channels (as in the Lightning Network). In this case they learn the predecessor, successor, and amount of a transaction they were involved in by simply comparing the balances on their channels before and after the transaction.

We only consider path finding protocols Π, which always find the cheapest path. Defining ideal privacy for protocols that do not necessarily output the cheapest path is more complex because the privacy loss depends on the discovered path. We also only consider passive adversaries, that is, an adversary can corrupt users and learn their internal state, but not make them deviate from honestly executing the protocol (e.g., by providing wrong or inconsistent input).

Our privacy notion is inspired by the Indistinguishability under Chosen Topology Attack (IND-CTA) security definition from work on topology-hiding multiparty computation [17], and it is defined as follows: We consider an adversary that initially chooses two networks and a transaction for each of them, and

also a subset of nodes to corrupt. We then require that given the view of the corrupted nodes after the path finding protocols has been executed on one of the two networks, an adversary cannot determine which. Of course we must require that the adversary chooses the networks, transactions, and corrupted nodes such that the corrupted nodes have the same neighbours and fee functions, and the final output of the corrupted nodes (either they are not on the path, and if, they learn their predecessor, successor and amount to be transferred) is identical in both cases; otherwise distinguishing between both networks is trivial for any protocol as one can distinguish using just the initial view and final output of the protocol. We give a more formal definition below.

3.1 The Ideal Privacy Security Game

We consider a security game involving an adversary $\mathcal{A}$ against a path-finding protocol Π. The protocol is run by the players V on a network $G = (V, E)$. Each player initially gets as inputs its neighbours and fee functions.

When the protocol starts, two players u_s, u_r get as extra input (u_s, u_r, x) informing them they are, respectively, the sender and the receiver of some amount x. The correctness we require from our protocol is that every $u \notin \mathsf{shortestPath}_{G^b}(u_s^b, u_r^b, x)$ outputs $\perp$, while every u on the path outputs its predecessor, successor and amount they transfer in this optimal path. The security game goes as follows:

- $\mathcal{A}$ chooses the following for $i \in \{0, 1\}$:
  1. A network (directed graph) $G^i = (V^i, E^i)$, where every edge (u, v) is labelled with a fee function $f^i_{u,v}(.)$.
  2. A sender and receiver pair (u_s^i, u_r^i) and amount x.

  $\mathcal{A}$ chooses a subset $S \subset V^0 \cap V^1$ of nodes to corrupt. These nodes must have the same neighbourhood and fee functions in both networks, and their final output (predecessor, successor and amount) must be identical.
- We choose a random bit $b \in \{0, 1\}$ and run Π on G^b (with input (u_s^b, u_r^b, x)).
- $\mathcal{A}$ gets the transcripts of the corrupted nodes.
- $\mathcal{A}$ outputs a bit b'. If $b' = b$, $\mathcal{A}$ wins the game.

Let us call a path finding protocol Π ϵ-private if $\mathcal{A}$ wins the above game with probability at most $1/2 + \epsilon$, and private if it is ϵ secure for some negligible ϵ.

3.2 Protocols with Ideal Privacy from MPC

If we assume a trusted third party $\mathcal{T}$ (that cannot be corrupted by the adversary and has a channel to every node in the network), the design of a private path-finding protocol is not challenging. In particular, each party in the network can send their data to $\mathcal{T}$, which will then locally compute the cheapest path and send the output, i.e., either $\perp$ or the amount, successor and predecessor in the cheapest path, to every party.

To design a protocol without a trusted third party, we can instantiate $\mathcal{T}$ using a multi-party computation (MPC) protocol. As here the users need to share pairwise channels, they need to know about the other users, which means in our security definition we need the users V^1 and V^2 in the two networks to be identical $V^1 \equiv V^2$. Our security notion is inspired by notions from topology-hiding MPC, where the goal is to hide the topology of the communication channels. Instead, we assume pairwise channels but want to hide the topology of the payment network. But of course we could also use a topology-hiding MPC to instantiate $\mathcal{T}$, in which case we would only require communication channels between users that share a channel. In this setting, one could potentially also achieve security when $V^1 \neq V^2$ as nodes would only talk to their neighbours, and for the corrupted nodes, these are identical.

3.3 A Practical Notion of Privacy

Here we outline the following game as well as some metrics to measure the privacy of any route discovery protocol over an arbitrary network. The game is played with an adversary $\mathcal{A}$ over an arbitrary network $G = (V, E)$. The adversary $\mathcal{A}$ is given some budget $B \in \mathbb{N}$ and $\mathcal{A}$ can corrupt (as defined in Section 3.1) any number of nodes in the network such that the total number of edges incident to these nodes is no greater than B. Here we emphasise two pertinent aspects of the corruption process: first, when $\mathcal{A}$ corrupts a node, *all* of the node's channels must be added to the total count, and second, we do not double count channels that have already been accounted for (which is the case where $\mathcal{A}$ corrupts two neighbouring nodes). The constraint on the number of edges the adversary can create captures the notion that creating new nodes (i.e., wallets) in a PCN is cheap, however, creating new edges (i.e., channels) comes with some fixed cost.

We denote by Π a route discovery protocol run by a pair of honest nodes (a source and a sink) over the graph G which may contain outputs (henceforth called messages) for both honest and adversarial nodes in the process of running the protocol. The goal of the adversary is to correctly identify the source or sink of Π upon receiving a set of messages from Π.

Estimators. Let $\mathcal{M}$ denote the space of all messages adversarial nodes may receive in process of honest nodes running Π. An *estimator* is simply a function g that takes as input a set of messages M such that each message in the set is from $\mathcal{M}$ and outputs a pair of nodes in V. That is, given a set of messages from Π, the adversary outputs a guess of the nodes that are the source and sink of the route finding protocol Π.

Privacy metrics. We adopt a similar approach as in [30] and use *recall* to measure the privacy our of route discovery protocols. Recall is a common performance metric used in information retrieval to evaluate estimators in classification settings. Let $M_{(u,v)}$ denote the set of messages the adversary receives that originates from a pair of honest nodes $u, v \in V$ running an instance of Π. The recall of an estimator g is defined as the number of classifications for the pair (u, v) where either u is classified as the source or v is classified as the sink

over the total number of instances where Π is run between u as source and v as sink. Formally,

$$Rec_{g,(u,v)} = \mathbb{1}\{g(M_{(u,v)}) = (u, \cdot) \vee g(M_{(u,v)}) = (\cdot, v)\} \tag{1}$$

We note that it is common to average as well as macro-average the recall over all honest nodes to get the macro-averaged expected recall $\mathbb{E}[Rec_g] = \frac{1}{|V \times V|} \sum_{(u,v) \in V \times V} \mathbb{E}[Rec_{g,(u,v)}]$.

4 A Family of Protocols

Consider a sender $u_s \in V$ who wants to transfer x coins to a receiver $u_r \in V$ and thus needs to know a path for the transaction, ideally the cheapest one. Next, we present a family of exploratory route discovery protocols that provide solutions to this problem. At its core, these protocols employ local probing: nodes send exploratory messages (originating at the sender and receiver) to their neighbours who in turn propagate them. Our protocols only require nodes to know their incident channels, and some also require a degree estimate of each neighbour.

The protocols run in three phases, (1) *exploration*, which runs until the first node receives both messages, i.e., the one originating at the sender and the receiver, (2) *notification*, where the relevant nodes are informed that a path was found and (3) *stopping*, where the nodes currently participating in the exploration phase are informed so they do not propagate messages further. Phase (1) is running slow, i.e., messages are propagated with some delay which should be significantly larger than the typical network delay, while in phase (2) and (3) messages are relayed immediately. The main reason we need Phase (1) to be slow is so the messages in the stopping phase can easily "catch up" to the nodes which are in the exploration phase. This further helps to improve correctness and even privacy.

Our protocols differ only on the proportion of nodes each node forwards the message to. On one extreme, we have Forward-to-All that involves nodes sending exploratory messages to all their neighbours, where each message is delayed for some time proportional to the fees. As a result, the optimal path is always discovered, and moreover, the first path that is found is also the cheapest one. On the other end of the spectrum, we have a more parsimonious protocol, namely Degree-Proportional Random Walk, that only involves sending messages to one neighbour. In this case, we expect fewer nodes to be involved in the discovery process but the optimal path may not be discovered.

As the protocols in the family are similar, we first present a generic overview of Forward-to-All, and then briefly describe how to modify it to get Degree-Proportional Random Walk. We then suggest some improvements to boost the privacy of this family of protocols. For clarity of exposition, we leave the detailed description of our protocols to Appendix A in our extended technical report [4].

4.1 Forward-to-All Exploration Phase

In this protocol, both the sender u_s and receiver u_r create messages with a special identifier (so intermediate nodes who receive messages from u_s and u_r can associate them together), an amount x that u_s wants to send to u_r, as well as a tag Sender or Receiver which specifies whether they are sending or receiving the transaction. The sender and receiver then propagate these messages through the graph by sending these messages to all their neighbours who then in turn propagate the message to all their neighbours. At every step of the propagation, the nodes update the transaction amount adding their desired fees for routing the transaction, and only forward the message after some time period has elapsed that is proportional to their desired fees. All nodes store the messages they received, as well as an id (not to be confused with identifier) of the node that sent them the message. The precise rule and fee computation differs, however, depending on whether a node gets a message from the sender or the receiver.

Fee computation for messages from receiver. Apart from the receiver, each intermediate node u_i upon receiving a message with the Receiver tag from another node u_{i+1}, updates the transaction amount to add a fee for sending the transaction amount along the channel (u_i, u_{i+1}). This is to reflect the fee u_i would charge for forwarding the transaction to u_{i+1}. Figure 1 illustrates this process where the receiver u_r sends a message with the transaction amount x to all of u_r's neighbours. Upon receiving the message from u_r, u_{i+1} adds a fee of $f_{u_{i+1},u_r}(x)$ to the transaction x. Messages with this updated transaction amount of $x + f_{u_{i+1},u_r}(x)$ would be sent to all of u_{i+1}'s neighbours.

Fee computation for messages from sender. The fee computation for the sender and the nodes that receive messages with the Sender tag is trickier. Although the sender knows the transaction amount x, they do not know the total amount they would have to send at the end of the protocol as it would include the fees along the path which is still unknown. Thus the sender would have to add an estimate of the total fee of the path, δ, to the transaction amount in their initial message. Each node that receives a message with the Sender tag, updates the transaction amount to subtract a fee for each edge they propagate the message to. This is to account for the fees the node will charge to forward the transaction. For instance in Figure 1, the node u_{i-1}, upon receiving a message with transaction amount $x + \delta$, subtracts $f_{u_{i-1},v}(x + \delta)$ from the transaction amount before forwarding the message with this new transaction amount to the node v. The node u_{i-1} does the same but subtracts $f_{u_{i-1},u_i}(x + \delta)$ from the transaction amount before sending the message to u_i.

Delay time computation. Let d be a publicly available delay function that maps fees to delay times. Let $d_{u,v}$ denote the delay time for the total fee for sending an arbitrary but fixed amount x over the channel (u, v), i.e., $d_{u,v} = d(f_{u,v}(x))$. Every node (except the sender and receiver) computes a delay time with the delay function d and the fees computed as described above. In Figure 1 for instance, since the sender u_s and receiver u_r do not have fees, u_s and u_r will send their exploratory messages immediately. The node u_{i+1} will wait d_{u_{i+1},u_r}

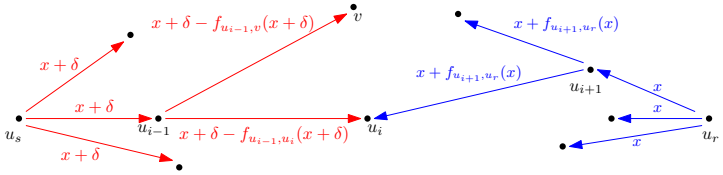


Fig. 1: Propagating exploratory messages from sender and receiver in the Forward-to-All protocol. Each directed edge (u, v) is labelled with the transaction amount in the message that u sends to v.

before forwarding the message to u_i, and u_{i-1} will wait d_{u_{i-1},u_i} before forwarding the message to u_i.

4.2 Forward-to-All Notification Phase

Upon receiving a new message, a node checks its identifier with the identifiers of the stored messages to see if the message identifiers can be associated together. When a node u_i finds an association of identifiers that indicate two messages are from the sender and receiver of a given sender-receiver pair, u_i begins a process of notifying both sender and receiver that a path exists between them. We denote the two nodes that sent u_i the associated sender and receiver messages by u_{i-1} and u_{i+1}, respectively. We also denote the transaction amounts in these messages as x_s and x_r respectively. Then, u_i immediately sends u_{i-1} a message with the identifier and amount x_s (resp. x_r to u_{i+1}). Both u_{i-1} and u_{i+1} then identify the nodes that sent them the messages with the same identifier and forward these messages to these nodes. Refer to Figure 2 for an illustration of the process.

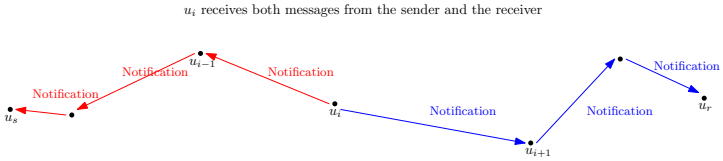


Fig. 2: Sending informative messages back to sender and receiver.

This process repeats itself until the sender and receiver get the message. At this point, the sender has enough information to proceed with the transaction. In particular, the sender can easily compute the total fee of the path from $x, x+\delta, x_s$ and x_r (communicated by the receiver).

Optimality of the discovered route. Using delay guarantees that in Forward-to-All, either the notification message corresponding to the shortest path (subject to the accuracy of the fee estimation of the sender) always reaches the sender

first, or the sender can find out if someone on the path deviated from the delay protocol. For example, let L^* be the optimal path from u_s to u_r and L' be a strictly more expensive path. Suppose several adversarial nodes on L' immediately forward messages without delay and as a result, u_s receives the notification message from an intermediate node on L' first. Since u_s knows the time they sent the first exploratory messages, u_s can extract the fees from the message received and check if the total delay time on this path is larger than the difference of the current time and the time u_s sent the first exploratory messages.

4.3 Forward-to-All Stopping Phase

When both sender and receiver are aware that a path exists between them and the sender is satisfied with the cost of the path, both sender and receiver can stop the protocol by sending a *stop message* with their identifiers to the nodes they sent the exploratory messages to. Nodes that have not yet sent the exploratory message to their neighbours would, upon receiving the stop message with the identifier, stop the message propagation. Nodes that have already send the exploratory messages would forward the stop message to the neighbours they sent the exploratory message to. This process is fast and thus will reach the slow propagation of the exploratory messages.

4.4 Degree-Proportional Random Walk

The Degree-Proportional Random Walk protocol is analagous to the Forward-to-All protocol with the exception that each node only forwards the message to *one* neighbour. Specifically, each node chooses a neighbour to forward the message to randomly with probability proportional to its degree. Thus, the messages are propagated according to two weighted random walks on the network, one starting from the sender and the other from the receiver, with the weight of any directed edge (u, v) corresponding to the degree of v. We observe that due to the probabilistic nature of Degree-Proportional Random Walk, optimality of the discovered path is not guaranteed unlike in Forward-to-All.

4.5 Improving Privacy

From the messages that originate at the sender and receiver and propagate through the network, we only need the property that one can efficiently recognise when a message from both is received. The simplest solution is to simply sample some random nonce I and propagate it together with a one bit tag specifying whether it is a sender or receiver originating message.

These messages are, however, completely linkable. This is unfortunate as it means that even if many path finding protocols are executed over the network at the same time, a potential adversary that controls some nodes in the network will still recognize with certainty which messages belong to the same path finding request. Thus, we do not leverage the fact that many protocols are running at the same time to improve privacy.

Making messages unlinkeable using bilinear maps. We can improve unlinkability by using a bilinear map [12] $e : G_1 \times G_2 \to G_T$ (such a map allows "for one multiplication in the exponent" as $e(g_1^a, g_2^b) = g_T^{a \cdot b}$ where g_1, g_2, g_T are generators of G_1, G_2, G_T) for a group where the DDH assumption holds in G_1 and G_2. Concretely, the sender and receiver sample a random x and then the sender, for every outgoing edge, samples a random r and propagates $(g_1^{x \cdot r} g_1^r)$ as the identifier, while the receiver propagates $(g_2^{r'/x}, g_2^{r'})$.

A node that receives $(g_1^{x \cdot r}, g_1^r)$ (similarly for the receiver tuples) propagates it only after re-randomizing it by exponentiating both elements with some fresh r' which gives a tuple $(g_1^{x \cdot r''}, g_1^{r''})$ where $r'' = r \cdot r'$. This way an adversary (who does not know x) will not be able to distinguish a pair of tuples of the form $(g_1^{x \cdot r}, g_1^r), (g_1^{x \cdot r'}, g_1^{r'})$ from random, and thus cannot decide whether they belong to the same instantiation of the path finding protocol. We stress that the unlinkeability is limited as it only holds if the adversary has access to messages originating either only at the sender or only at the receiver. This is inherent as we need parties who receive tuples (a, b) and (a', b') originating at both to efficiently recognize a path is found by checking whether $e(a, a') = e(b, b')$ as

$$(a, b) = (g_1^{x \cdot r}, g_1^r), (a', b') = (g_2^{r'/x}, g_2^{r'}) \Rightarrow e(a, a') = e(b, b') = g_T^{r \cdot r'}$$

Quantising the transaction and encrypting the fees. Messages that contain the exact transaction amount are also linkable, even when fees are added, as the fees are typically miniscule compared to the transaction amount. To reduce this linkability (at the cost of fee accuracy), the sender and receiver can quantise the amount of the transaction by rounding it up to a predefined value (for instance, a power of 2). Then, instead of adding the fees to the quantised transaction amount, nodes encrypt their fees using an additive homomorphic encryption scheme such as additive ElGamal encryption. If several protocols are running simultaneously on the network, nodes will see many messages with the same quantised transaction amount, effectively reducing linkeability.

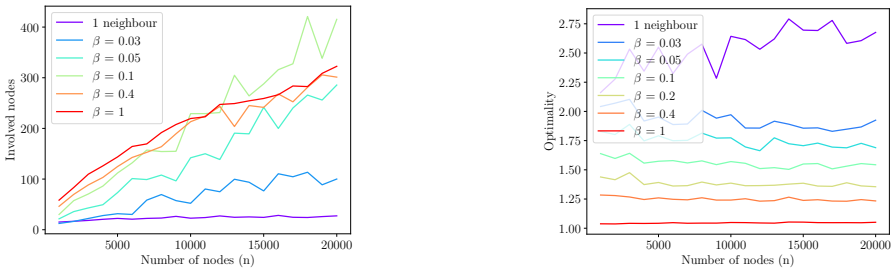
5 Analysis and Evaluation

In this section, we empirically study the *efficiency, optimality* and *privacy* of our family of protocols described in Section 4 on a recent snapshot (September 2021) of the Lightning network [9], which comprises of 13780 nodes and 63518 channels. We parameterise our family of protocols by $\beta \in (0, 1]$ which represents the proportion (rounded up to the nearest integer) of neighbouring nodes in the Degree-Proportional Random Walk protocol that any node chooses to pass messages to. We note that $\beta = 1$ corresponds to Forward-to-All. In our experiments using the Lightning network, we chose 1000 random pairs of sender and receivers and ran our protocols for selected values of β with these random pairs. Our choices of β can be seen in Table 1 which also presents a summary of our results.

5.1 Efficiency

We measure the efficiency of our protocols by the communication overhead. This is measured by the average number of involved nodes in one route discovery attempt, i.e., the number of nodes that receive at least one message in any given run of our protocol. As we can see from the second column of Table 1, the expected number of involved nodes in a single route discovery attempt ranges from 16 (0.1% of the network) to 459 (3% of the network) with the smallest value corresponding to sending messages to just 1 neighbour and the largest value corresponding to Forward-to-All.

We are also interested to see how our protocols perform as the Lightning Network scales in size. To do so, we perform simulations of our protocols on Barabási–Albert (BA) graphs. The BA model [7] is a popular algorithm to create scale free networks using a preferential attachment mechanism. Many real world networks, including the Lightning network, are characterized as scale-free [6,22]. Figure 3 presents the average communication overhead as well as the average ratio of the length of the found path over the optimal path. Our empirical simulations show that the communication overhead for the BA graph with 20000 nodes is similarly low (2%) when compared to the Lightning Network. We finally complement our empirical analysis with a theorem (proof in Appendix D of [4]) which states that for BA graphs the communication overhead of the Degree-Proportional Random Walk protocol scales sublinearly in the number of nodes n.



(a) Communication overhead

(b) Ratio of length of found path to length of optimal path

Fig. 3: Efficiency and optimality of our routing algorithms on Barabasi-Albert graphs

Theorem 1. *The expected number of involved nodes in the Degree-Proportional Random Walk protocol on a BA graph with n nodes is $O(\sqrt{n} \cdot \frac{\log^2 n}{\log \log n})$.*

5.2 Optimality

The fee-proportional delay function that we described in the Notification Phase of Section 4 guarantees that the first message that reaches the sender and the receiver corresponds to the shortest path in Forward-to-All. There is no such guarantee, however, for our protocols in the case where $\beta < 1$ due to its probabilistic nature. To measure the optimality of the discovered path for our protocols when $\beta < 1$, we look at the average ratio of the discovered path length over the length of the shortest path for various choices of β in the 1000 runs as described above. In the third column of Table 1, we observe that protocols with $\beta < 1$ return longer paths on average compared to Forward-to-All, as expected. However, even for small values of β, e.g., $\beta = 0.01$, the average ratio is no more than 2.

β	# involved nodes	$\frac{len(\text{found path})}{len(\text{shortest path})}$	Recall
only 1 neighbour	16	3.36	0.129
1%	34	1.54	0.129
10%	86	1.14	0.191
20%	133	1.10	0.249
40%	277	1.07	0.29
80%	431	1.01	0.371
100%	459	1	0.43

Table 1: Summary of the efficiency, optimality and privacy of our protocols on the Lightning Network, run with budget 0.15 in our estimation of recall.

5.3 Privacy

We empirically measure the privacy of our protocols using the notion of recall as defined in Section 3.3. We stress that there are two integral components of our notion of recall. The first is the corruption strategy of the adversary, which is used by the adversary to choose nodes up to the corruption budget B to corrupt. The second is the choice of estimator the adversary uses to guess the source and sink of an instance of the route discovery protocol. We will first describe 3 corruption strategies, and then define the first spy estimator. We further show that the first spy estimator is in general a good guessing heuristic for the adversary.

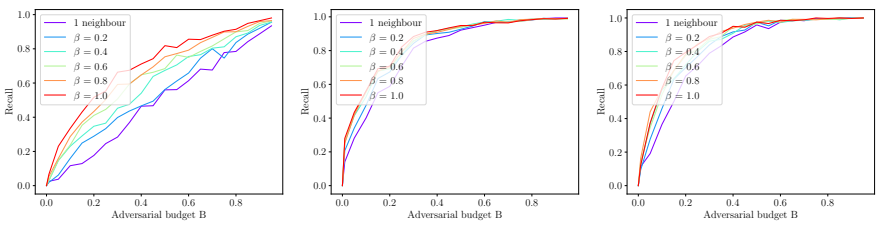
Corruption strategies. The first strategy, called *random corruption*, involves choosing nodes at random in the network to corrupt (and all their edges) until the total number of corrupted edges is less the adversarial budget B. In the second strategy, termed *well-connected corruption*, we first sort the nodes in the network based on their degree and then sequentially corrupt nodes in descending order to their degree until the budget is depleted. The intuition behind this strategy is

that high-degree nodes tend to serve as hubs that route a large proportion of the transactions in the Lightning Network [9,1]. However, this strategy assumes the adversary has *full* knowledge of the degree of all nodes in the network, which is not the case in private PCNs. To achieve similar results with only local knowledge of the graph, we introduce our third strategy, called *random hub corruption*. In this strategy, the adversary starts with a random node, but instead of corrupting the node, it randomly corrupts one of its neighbours. This process continues until the budget is depleted. Intuitively, the majority of nodes in a PCN are users that are connected to high-degree hub nodes. As such, there is a high chance that a random neighbour of a selected node is a hub node. An algorithmic description of all 3 corruption strategies is given in Appendix B.1 in our extended version [4].

First spy estimator. The choice of estimator in our experiments is the *first-spy estimator* [30], which is simply guessing the first honest node that passes a message from the protocol to any adversarial node as the source. We justify our choice of estimator with the following lemma (proof in Appendix B.2 in our extended version [4]) which shows that the first spy estimator is optimal in a restricted setting where messages are not re-randomised, sender and receivers are randomly chosen, and timing assumptions are not taken into account.

Lemma 1. *For Degree-Proportional Random Walk (without re-randomisation), when sender-receiver pairs are chosen uniformly and independently from honest nodes and both propagate their messages independently and randomly, the first spy estimator is the Maximum a Posteriori (MAP) estimator.*

We stress that our theoretical results hold in a restricted setting which does not mirror the realistic setting which we run our experiments on. Indeed in Appendix B.4 in our full paper [4], we highlight some limitations to the first spy estimator under more realistic assumptions. Nevertheless, we believe it is a good first step guessing heuristic and we leave developing stronger theoretical results in the realistic setting as an interesting direction of future work.



(a) Random (b) Well-connected (c) Random hub

Fig. 4: Average recall over 1000 random runs of the protocols in the Lightning network for different corruption strategies

Average recall. Figure 4 presents the average recall for each of the corruption strategies given the first spy estimator over 1000 runs of each protocol. The well-connected and random hub corruption strategies perform strictly better in terms of recall for all β values compared to random corruption. Moreover, random hub corruption performs almost equally well compared to well-connected corruption and thus is a good choice to use in practice when there is only partial information known about node degrees. Finally, our plots show that the adversary achieves the highest average recall with Forward-to-All and the lowest with Degree-Proportional Random Walk.

Optimality and efficiency/privacy trade-off. In Forward-to-All, nodes forward messages to all their neighbours, and as a result a large fraction of the network is involved in the route-discovery process, which as we see in Section 5.1 and Section 5.3 has negative impact on the efficiency and privacy of the protocol. On the other hand, reaching every node is the only way to guarantee the shortest path is always found. In Degree-Proportional Random Walk, each node just forwards messages to one neighbour and thus only a very small fraction of the graph is involved. However, as we see from Section 5.1 the paths are longer on average. We note that β values in between these extremes allow users to trade off between optimality and efficiency/privacy.

6 Related Work

Existing work on route discovery in PCNs can be broadly classified into two categories: solutions which focus on efficiency, and solutions which focus on privacy.

On the efficiency front, Flare [21] and SilentWhispers [15] route payments through highly connected nodes to improve the scalability of route discovery. SpeedyMurmurs [24] and VOUTE [23] employ a similar routing technique called prefix embeddings, which makes the process even faster. These solutions require nodes to have global knowledge of the network, whereas we present a protocol that does not require any knowledge of the PCN topology. Spider Network [25] splits payments into smaller units and routes them over multiple paths using waterfilling. However, this does not guarantee the discovery of an optimal path, whereas our protocol guarantees optimality by adding a fee-proportional delay in the route discovery process. Flash [31] uses a modified max-flow algorithm to find the optimal path, but also requires nodes to have global knowledge of the network. Perun [2] avoids routing through intermediaries altogether by introducing the notion of virtual channels. However, this does not solve the route discovery problem.

On the privacy front, MAPPCN [29] focuses on anonymity and privacy during transaction execution, but does not address the issue of route discovery as users are required to know the payment path. LightPIR [19] uses private information retrieval to perform private route discovery efficiently, but does not account for optimality in the case of private channels. In contrast, our protocols employ local probing, thus our solutions are still optimal even with private channels. Recently, [27] uses MPC to perform privacy preserving routing of transactions,

however it is only limited to fixed star graph topologies. Finally, [14,13] present routing protocols that also do not assume information about the topology of the involved PCN, but lack formal definitions of privacy and evaluation metrics.

7 Conclusion

We presented route discovery protocols that are suitable for private PCNs. We first formalized the ideal notion of privacy in PCNs, and showed that ideal privacy is feasible yet inefficient. We then presented a family of practical route discovery protocols which trade off between optimality and efficiency/privacy. To evaluate their privacy leakage, we introduced and leveraged a novel practical notion of privacy.

The simulation of our protocols on the Lighting Network and Barabási–Albert graphs validates our approach, unveiling the aforementioned trade-off. We also observe that our protocols involve a very small fraction of the network on average, showcasing we can indeed design efficient and private routing algorithms that rely on minimal to no assumptions on the topology of the PCN with almost optimal results. From our simulations on Lightning we further deduce that an effective strategy for an adversary is to connect with high-degree nodes, i.e., payment hubs. We also discover through our empirical simulations on large Barabási–Albert graphs that the efficiency and privacy of our algorithms perform much better on average compared to our theoretical upper bound, which demonstrates that our algorithms also scale efficiently with the size of PCNs.

Acknowledgements. This work was supported in part by the ERC CoG 863818 (ForM-SMArt), Austrian Science Fund (FWF) 10.55776/COE12, and MOE-T2EP20122-0014 (Data-Driven Distributed Algorithms) grants.

References

1. Lightning network search and analysis engine. `1ml.com`
2. Perun: Virtual payment hubs over cryptocurrencies. In: IEEE S&P (2017)
3. Akavia, A., LaVigne, R., Moran, T.: Topology-hiding computation on all graphs. J. Cryptology pp. 176–227 (2020)
4. Avarikioti, Z., Bastankhah, M., Maddah-Ali, M.A., Pietrzak, K., Svoboda, J., Yeo, M.: Route discovery in private payment channel networks. IACR Cryptol. ePrint Arch. p. 1539 (2021)
5. Avarikioti, Z., Kogias, E.K., Wattenhofer, R., Zindros, D.: Brick: Asynchronous incentive-compatible payment channels. In: FC (2021)
6. Barabási, A.L., Albert, R., Jeong, H.: Scale-free characteristics of random networks: the topology of the world-wide web. Physica A: statistical mechanics and its applications pp. 69–77 (2000)
7. Barabási, A.L., Pósfai, M.: Network science. Cambridge University Press, Cambridge (2016), `http://barabasi.com/networksciencebook/`
8. Bitcoin community: Bitcoin core 0.21.0-based taproot client 0.1. `https://bitcointaproot.cc/` (2021)

9. Decker, C.: Lightning network research; topology, datasets. `https://github.com/lnresearch/topology`, accessed: 2022-10-01
10. Decker, C., Russell, R., Osuntokun, O.: eltoo: A simple layer2 protocol for bitcoin. `https://blockstream.com/eltoo.pdf` (2018)
11. Decker, C., Wattenhofer, R.: A fast and scalable payment network with bitcoin duplex micropayment channels. In: Stabilization, Safety, and Security of Distributed Systems. pp. 3–18. Springer (2015)
12. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics **156**(16), 3113–3121 (2008)
13. Grunspan, C., Lehéricy, G., Pérez-Marco, R.: Ant routing scalability for the lightning network. CoRR **abs/2002.01374** (2020)
14. Grunspan, C., Pérez-Marco, R.: Ant routing algorithm for the lightning network. CoRR **abs/1807.00151** (2018)
15. Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M.: Silentwhispers: Enforcing security and privacy in decentralized credit networks. In: NDSS (2017)
16. Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple schnorr multi-signatures with applications to bitcoin. Des. Codes Cryptogr. **87**(9), 2139–2164 (2019)
17. Moran, T., Orlov, I., Richelson, S.: Topology-hiding computation. In: Theory of Cryptography Conference. pp. 159–181. Springer (2015)
18. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
19. Pietrzak, K., Salem, I., Schmid, S., Yeo, M.: Lightpir: Privacy-preserving route discovery for payment channel networks. In: Proc. IFIP Networking (2021)
20. Poon, J., Dryja, T.: The bitcoin lightning network: Scalable off-chain instant payments (2016)
21. Prihodko, P., Zhigulin, S., Sahno, M., Ostrovskiy, A., Osuntokun, O.: Flare: An approach to routing in lightning network. White Paper (2016)
22. Rohrer, E., Malliaris, J., Tschorsch, F.: Discharged payment channels: Quantifying the lightning network's resilience to topology-based attacks. In: 2019 IEEE EuroS&P Workshop. IEEE (2019)
23. Roos, S., Beck, M., Strufe, T.: Voute-virtual overlays using tree embeddings. arXiv: 1601.06119 (2016)
24. Roos, S., Moreno-Sanchez, P., Kate, A., Goldberg, I.: Settling payments fast and private: Efficient decentralized routing for path-based transactions. arXiv: 1709.05748 (2017)
25. Sivaraman, V., Venkatakrishnan, S.B., Alizadeh, M., Fanti, G., Viswanath, P.: Routing cryptocurrency with the spider network. In: Proc. 17th ACM Workshop on Hot Topics in Networks. pp. 29–35 (2018)
26. Spilman, J.: Anti dos for tx replacement. `https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2013-April/002433.html`, accessed: 2020-11-22
27. Tiwari, S., Yeo, M., Avarikioti, Z., Salem, I., Pietrzak, K., Schmid, S.: Wiser: Increasing throughput in payment channel networks with transaction aggregation. CoRR **abs/2205.11597** (2022)
28. Tochner, S., Zohar, A., Schmid, S.: Route hijacking and dos in off-chain networks. In: AFT (2020)
29. Tripathy, S., Mohanty, S.K.: Mappcn: Multi-hop anonymous and privacy-preserving payment channel network. In: FC. pp. 481–495. Springer (2020)
30. Venkatakrishnan, S.B., Fanti, G., Viswanath, P.: Dandelion: Redesigning the bitcoin network for anonymity. Proc. ACM Meas. Anal. Comput. Syst. **1**(1) (2017)
31. Wang, P., Xu, H., Jin, X., Wang, T.: Flash: Efficient dynamic routing for offchain networks. In: International Conference on Emerging Networking Experiments And Technologies. p. 370–381 (2019)

A comparative study of Rust smart contract SDKs for Application-Specific Blockchains

Jan Vanhoof[0009−0002−1336−5765] and Tom Van Cutsem[0000−0003−4116−4290]

DistriNet, KU Leuven
jan.vanhoof1@kuleuven.be
tom.vancutsem@kuleuven.be

Abstract. This paper reports on a comparative study of two recent software development kits (SDKs) for so-called "Application-specific Blockchains" that support development of smart contracts in general-purpose languages. Specifically we report on the similarities and differences between Cosmwasm and Polkadot smart contracts written in the Rust programming language.
To help guide our comparative study we start from a representative set of Solidity smart contracts, namely an English auction contract and a Non-Fungible Token (NFT) contract. Both contracts offer insights into the requirements that must be offered by a smart contract SDK. We develop a concrete baseline for comparison between Cosmwasm and Polkadot by translating the two Solidity contracts into Rust, using the respective SDKs.
Our comparison defines a starting point for better understanding the design space of smart contract SDKs, and what the advantages and disadvantages are of different API interfaces and execution models.

Keywords: Smart contracts · API design · Rust

1 Introduction

The Ethereum [5] blockchain pioneered the concept of a "programmable" blockchain by allowing users to deploy small programs on the blockchain, so-called *smart contracts*. Such smart contracts have been widely adopted for use cases ranging from creating custom (fungible and non-fungible) tokens, auctions, decentralized exchanges and community-controlled token vaults (DAOs).

Despite the widespread adoption, smart contracts in Ethereum remain mostly limited to small-scale programs (both in program size as well as in runtime execution). The most widely used smart contract language, Solidity, is a domain-specific language designed specifically to write logic to execute on the Ethereum Virtual Machine (EVM). Despite its domain-specific design, programs written in Solidity are prone to a wide range of vulnerabilities, many of which have led to large financial losses [2].

In addition, Ethereum has been a victim of its own success, with network congestion leading to more and more smart contracts competing for limited

transaction throughput and block storage, leading to large "gas fees" (transaction costs), rendering many use cases too expensive to run as contracts on the blockchain.

In response to these issues, *application-specific* blockchains, also called "Appchains", have arisen. In contrast to Ethereum where all contracts share a single common blockchain, appchains support multiple independent blockchains that each support one, or a few, application use cases. To avoid weakened security or interoperability, most appchain platforms offer built-in ways for these separate chains to interconnect, enabling shared security and transferability across chains, to varying degrees.

Another distinguishing feature of appchain platforms compared to Ethereum and its EVM, is their support for a general-purpose runtime environment to host and execute smart contract code. Here, WebAssembly [9] is a widely used choice because of its performance, portability across operating systems and instruction sets, sandboxed execution environment, and common compilation target for a large number of source languages.

WebAssembly is a portable bytecode format, but does not natively support the features required to implement smart contracts and to interact with a blockchain environment. Appchains must thus make an API available to the WebAssembly code, and this API is usually exposed to the higher-level source language in the form of a Software Development Kit (SDK) that helps developers design, deploy and debug high-level contract code. The Rust programming language is a widely adopted language for this purpose, because of its combination of generating efficient code, small binaries, and expressive type system and memory-safe programming model.

In this paper we perform a comparative study of the execution layer for the two most popular appchain platforms, namely Cosmwasm and Polkadot.[1] Both appchains offer a Rust SDK for developing WebAssembly-based smart contracts (Section 2). To help guide our comparative study we start from a representative set of Solidity smart contracts, namely an English auction contract and a Non-Fungible Token (NFT) contract (Section 3). Both contracts offer insights into the requirements that must be offered by a smart contract SDK (Section 4). We develop a concrete baseline for comparison between Cosmwasm and Polkadot by translating the two Solidity contracts into Rust, using the respective appchain SDKs (Section 5). Our comparison provides a starting point for better understanding the design space of smart contract SDKs, and what the advantages and disadvantages are of different API interfaces and execution models (Section 6).

[1] One metric for popularity is to look at active contributors to code repositories on GitHub, linked to appchain-related projects. According to Electric Capital's yearly Web3 Developer Report at `https://www.developerreport.com/` the top four projects based on total number of contributors on Dec. 31, 2023 were Ethereum, Polkadot, Polygon and Cosmos. Of these four only Polkadot and Cosmos are appchain projects (Polygon is an Ethereum side-chain and reuses the EVM for compatibility).

2 Background on Application-Specific Blockchains

Figure 1 offers a high-level overview of how smart contracts are compiled and deployed in Ethereum, Cosmwasm and Polkadot.

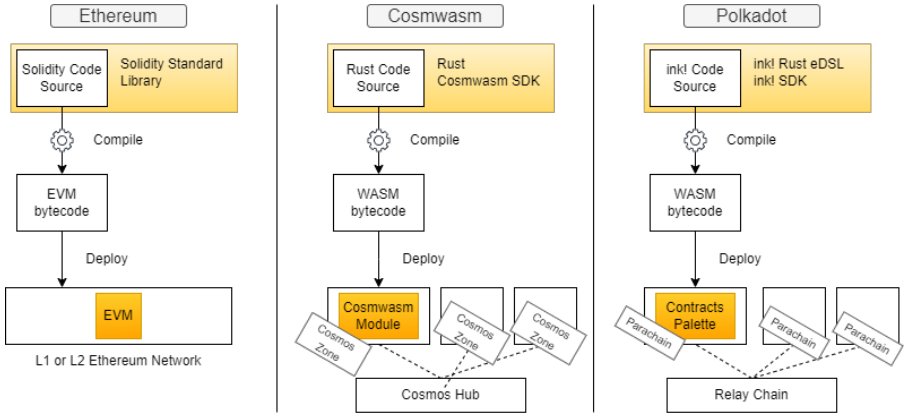


Fig. 1. Overview of smart contract compilation and deployment across different blockchains

Cosmos [10] is a network of interoperable blockchains, called Cosmos zones, built on the CometBFT consensus protocol. The project also offers the Cosmos SDK to allow blockchain developers to easily set up their own blockchain. Cosmwasm is a separate open source project to add support for WebAssembly-based smart contracts to any blockchain built with the Cosmos SDK. In addition, Cosmwasm is also the name of a public blockchain with built-in support for deploying WebAssembly smart contracts, similar to the public Ethereum network. Cosmos zones can communicate via a Cosmos hub, a central Cosmos network designed to relay messages, by using the Inter Blockchain Protocol (IBC).

Polkadot [14] is a multi-chain blockchain platform that offers a "relay chain" to which custom appchains, called "parachains", can be connected. Through this relay chain, parachains can communicate with each other via Cross-Chain Messages (XCM). To be allowed to connect to the relay chain, each parachain needs to participate in an auction and bid to get a slot as these are limited. Nodes of the relay chain will also serve as validators for the parachain to increase security. A custom Polkadot runtime can be composed of palettes (similar to modules). One of those palettes, the contracts palette, supports ink!, an embedded domain-specific language (eDSL) in Rust that compiles to WebAssembly.

3 Running example: NFT Auction

To be able to compare differences between the appchains, we will use two representative contracts, as most real world use cases don't consist of just one smart contract, but multiple which call each other. The first is an NFT Contract (ERC721), the second an NFT auction (English Auction), both based on examples from Solidity by Example [12]. The interaction between the two contracts and their participants is visualized in Figure 2.

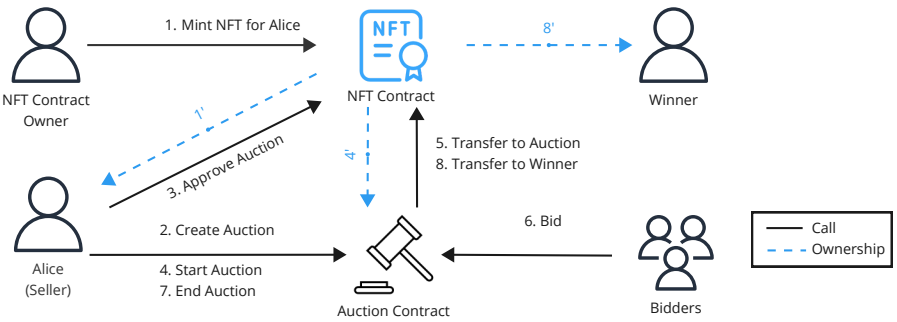


Fig. 2. Interaction Diagram of an NFT auction example

The interaction flow is as follows: *1.* The NFT Contract Owner Mints a new NFT for Alice. *2.* Alice creates a new NFT Auction Contract to sell the NFT. *3.* Alice adds the NFT Auction Contract as approver of the NFT. *4.* Alice starts the auction which triggers *5.*, the NFT Auction contract makes itself, as approver, the new owner of the NFT. *6.* Multiple bidders make a bid for the NFT. *7.* Alice ends the auction when the deadline has been reached which triggers *8.*, the NFT is transferred to the Winner (the person with the highest bid) and Alice gets the highest bid transferred to her account (not visualized).

3.1 The NFT Contract

For the NFT contract we use a simplified NFT contract based on the ERC721 standard [8]. Anyone can instantiate a new NFT Contract and becomes automatically the NFT contract owner. Only the contract owner is allowed to `Mint` new NFTs for a specified NFT owner. The NFT owner can allow another party to manage this specific NFT via the `Approve` action. Either the NFT owner or the approved party can transfer the ownership of the NFT to someone else with the `Transfer` action. Anyone can query the smart contract to check who the owner is of a specific NFT (`OwnerOf`), which NFTs belong to a certain owner (`BalanceOf`) and can request who the approver is for an NFT (`Approved`).

The NFT contract and its available methods are represented in Table 1.

Table 1. NFT Contract Methods

Action	Role	Conditions (NFT Contract nc, NFT n, Caller c)
Instantiate	Contract Owner	{}
Mint	Contract Owner	{nc.Owner == c}
Approve	Seller	{n.Owner == c}
Transfer	Seller or Approver	{n.Owner == c OR c ∈ n.Approved}
OwnerOf	Anyone	{}
BalanceOf	Anyone	{}
Approved	Anyone	{}

3.2 The Auction Contract

The second smart contract is an English auction. An NFT owner (Seller) can create a new auction and becomes owner of the contract. After granting approval to the NFT contract to manage the NFT, the seller can `Start` the auction. At this moment the end time of the auction is determined and the auction contract will take ownership of the NFT until the auction ends. Once started, everyone can bid on the NFT using the `Bid` method with funds attached. A bid is accepted if it is higher than the current highest bid and the auction has not ended. Previous bidders, except the current highest bidder, can `Withdraw` their previous bids. When the set amount of time has passed, anyone can `End` the auction. The contract will check if the minimum bid, set at creation, was reached and if so it transfers the NFT to the highest bidder and the seller is payed out. If the minimum bid was not reached, the NFT is returned to the seller.

Table 2 summarizes the methods provided by the NFT Auction contract.

Table 2. NFT Auction Contract Methods

Action	Role	Conditions (Auction a, NFT n, Caller c, Bid b)
Instantiate	Seller	{}
Start	Seller	{a.Owner == c AND a ∈ n.Approved)}
Bid	Anyone	{a.Started AND a.EndTime > now AND b.Value > a.StartBid}
Withdraw	Anyone	{c != a.HighestBidder}
End	Anyone	{a.Started AND a.EndTime <= now}

4 Smart Contract Framework Requirements

To be able to implement the smart contracts defined in Section 3, the appchain platform needs to fulfill certain requirements in its smart contract framework. We will give an overview of these requirements in this section.

Message Endpoints Each contract requires at least one accessible message endpoint for users and other contracts to interact with it.

Identification A user or contract needs to be identifiable to be able to link their digital assets (fungible or non-fungible) to them. Most blockchains represent user or contract identity with an address, based on the public key of the user or contract. Identification is used in the example contracts to assign ownership, transfer funds and for the auction contract to call the NFT contract.

Ownership Different digital assets, like fungible tokens, non-fungible tokens, but also smart contracts can be owned by a user or contract. This implies that only that user or contract has the exclusive authority to transfer or use the assets. In the auction example, ownership is used to determine who owns the smart contracts, who owns the NFT and for the balance of native tokens of each party.

Access Control Some actions can only be performed by users who fulfill certain conditions, e.g., the user needs to be owner or approver of a token to transfer it. How can a smart contract enforce this? In the example only the NFT owner and the NFT approver are allowed to transfer the NFT.

Transactionality In many situations a set of actions must either all succeed or all fail. How does the framework enable a smart contract to enforce this kind of atomicity rules? At the end of the auction, the NFT is sent to highest bidder and the funds are transferred to the seller. If either of these actions should fail, both action are reverted.

Persisting State Most smart contracts need to be able to maintain a persistent state across different function calls. State is required in the auction example in many cases, for example to keep track of which NFTs there are and who owns them. Or who placed already a bid in the auction and for what value.

Token Transfer Blockchains typically have a cryptocurrency, which is the native asset which can be traded with other users. This can be used to pay for a service like the execution of a contract or to buy a digital asset. There are also non-native tokens, which are non-unique (fungible) and can be exchanged in a similar way. The example makes use of native tokens to place a bid in the auction and to pay out to the seller.

Contract Metadata A contract needs to be able to acquire some information about itself relative to the blockchain which is not maintained in its own state. Examples of this are the contract address and the token balance of the contract. The auction contract needs to know its own address to transfer the NFT or transfer native tokens.

Blockchain State Smart contracts must be able to inspect the current state of the blockchain, like the current block height, look up previous transactions or access some other functionality offered by the blockchain like address validation. Address validation is required in the example to validate addresses passed in function parameters.

Notion of Time This can be considered as part of the Blockchain State but it deserves some extra attention. Real world use cases require a notion of time. A smart contract could base itself on the current block height, but that's not always an accurate measurement. Therefore most blockchains encode the current real world time as a timestamp in the blocks. This notion of time is required

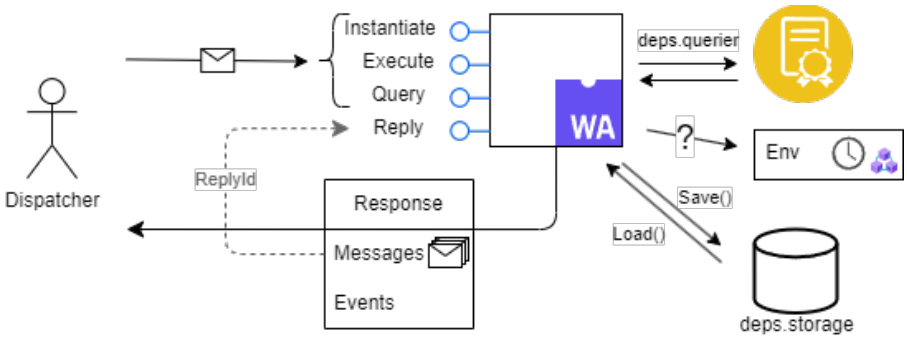


Fig. 3. Key interfaces of a Cosmwasm contract

in the NFT auction contract to be able to determine whether the auction has ended.

Call Smart Contracts Smart contracts need to be able to interact with other contracts for many different reasons, size limitations of a contract, dynamic adaptability, interacting with already existing contracts, ... By calling a smart contract we mean executing a method on the smart contract which changes the state of the contract. The auction smart contract needs to be able to call the NFT smart contract to transfer the NFT for which the auction was created.

Query Smart Contracts Similar to calling smart contracts, a contract needs to be able to query the current state of an existing contract. A query will not change the state of the other contract. This can be used by the auction contract to check whether the NFT is actually owned by the seller.

Events The decentralized application (dApp) ecosystem consists of more than components on the blockchain. To communicate with external applications, smart contracts emit events to notify that something noteworthy has happened.

5 Analysis of Appchain Rust SDKs based on smart contract requirements

To be able to understand how different appchain SDKs support the requirements that we put forward in Section 4, we implemented the NFT Auction presented in Section 3 in both Cosmwasm and Polkadot.[2] In this section we will elaborate on how each technology handles smart contracts with its provided smart contract SDK.

5.1 Smart contracts in Cosmwasm

Figure 3 offers an overview of the key interfaces for a smart contract in Cosmwasm.

[2] The source code of the smart contracts is available at `https://github.com/JvHKuL/RustSDKCompare`.

Cosmwasm is a module that can be used in any Cosmos appchain, but there already exists an appchain with the same name, Cosmwasm, that allows you to deploy your own smart contracts on a publicly hosted shared blockchain, similar to e.g. the Ethereum mainnet. At the moment Rust is the main programming language for Cosmwasm. Although the normal Rust compiler is used to compile the Rust code to WebAssembly, Cosmwasm does impose certain interface requirements for it to be compatible with the module. Cosmwasm offers different Rust crates to facilitate this setup and a `cosmwasm-check` utility to validate the compatibility of the resulting contract.

Listing 1.1. Cosmwasm contract entry point signatures

```
#[entry_point]
pub fn instantiate (Depsmut, Env, MessageInfo, InitMsg) ->
    StdResult<Response>
#[entry_point]
pub fn execute (DepsMut, Env, MessageInfo, ExecuteMsg) ->
    StdResult<Response>
#[entry_point]
pub fn query (Deps, Env, QueryMsg) -> StdResult<Binary>
#[entry_point]
pub fn reply(DepsMut, Env, Reply) -> StdResult<Response>
```

Each contract exposes so called "entry points" for the appchain to interact with it. Cosmwasm defines four basic entry points, `instantiate`, `execute`, `query` and `reply` (Listing 1.1). A valid Cosmwasm smart contract expects at least an `instantiate` entry point.

In Cosmwasm the bytecode of a contract is first uploaded to the blockchain. This code can then be instantiated via the `instantiate` entry point, resulting in a new instance of the contract with its own address and state.

The `execute` function is used to execute smart contract code that can alter the state of the contract or blockchain.

The `query` function is used to query the state of the smart contract or blockchain, without altering it.

`Deps(Mut)` is a utility parameter to communicate with the outer world. It allows querying and updating the contract state, querying (not changing) other contracts' state, and gives access to an API object with a couple of helper functions for dealing with Cosmwasm addresses.

`Env` represents the blockchain state when the message is executed. It contains the chain height and id, current timestamp, and the address of the called contract.

`MessageInfo` contains the address of caller and the native tokens that were sent with the request.

All functions return a `StdResult<Response>` returning either a value of type `Response` or an `Error`.

Contrary to Solidity, Cosmwasm doesn't allow the exposure of custom functions as entry point. Instead Custom Message types are used to make the distinction between what action needs to be triggered on the contract. This forces the developer to declare a new message type for each functionality the smart

contract needs to expose and avoids accidental exposure of functions. A message enters the contract via one of the three basic entry points and is then dispatched to the proper internal function based on the received message type.

Cosmwasm was designed this way because it follows the actor model [1]. Instead of directly executing calls on other contracts or appchain modules, a contract must return a message via the response to a dispatcher, which resides in the appchain. The dispatcher will then send the message to the proper contract or module. The resulting response will then be relayed back to the original contract. This approach has multiple advantages. There is loose coupling between the contracts: only the exchanged data format, the message, needs to be agreed upon. It avoids one of the most common problems in Solidity smart contracts, the re-entrancy attack [2], by only sequentially treating calls in a contract, guaranteeing that its state is finalized before a new message is processed. Cosmwasm assures the atomic execution of the messages and their submessages by creating a save point when a new external transaction is processed and performing optimistic updates. If all goes well, the transaction is committed, if it fails, a rollback is performed.

We will now analyse how the Cosmwasm SDK satisfies the requirements offered in Section 4.

Message Endpoints Cosmwasm offers predefined message endpoints, called "entry points", primarily `initiate`, `execute`, `query` and `reply`. Rust enumerations are used to distinguish different actions to be performed.

Identification Users and contracts are identified by their address. They have a predefined prefix based on the appchain.

Ownership A distinction needs to be made between the ownership of native tokens, a contract, or digital assets like NFTs. Native token ownership is handled by a balance per account (address). For contract ownership the `cw-ownable` utility crate provides macros to extend `execute` and `query` messages with default values and functions to set and validate the current ownership of a contract. Digital asset ownership is managed by the contract itself, typically by maintaining a mapping between the owner address and the asset.

Access Control Access control for the owner is done via the previously mentioned `cw-ownable` utility functions, specifically `cw_ownable::assert_owner`, which will throw a predefined `OwnershipError`. For other roles this is done by the contract itself with conditional statements and internally stored role information. In both cases the contract gets the caller information from the `MessageInfo` parameter of the called function.

Transactionality Atomicity is guaranteed by the hosting environment via the optimistic update and rollback mechanism. If a contract calls another contract, and that call fails, the main contract can decide whether its state is still committed or not. If there is a failure in the main contract, the state of both the main contract and all called contracts is rolled back.

Persisting State Cosmwasm advises to use the `cw-storage-plus` package which offers different classes that offer an abstraction on top of the standard `Storage` class. It offers an `Item` class to store basic types or `structs` and also a

`Map`. Important to note is that the developer is responsible for the loading and saving of these classes to and from persistent storage. However there are a variety of functions provided to hide this, e.g. a closure can be used to update a value in the storage with loading and saving done automatically. (See Listing 1.2)

Listing 1.2. Cosmwasm storage handling using a closure

```
pub owned_tokens_count: Map<'a, &'a Addr, u32>
owned_tokens_count: Map::new(owned_tokens_count_key)
...
self.owned_tokens_count
    .update(deps.storage, &owner_addr, |old| match old {
        Some(x) => Ok::<u32, ContractError>(x + 1),
        None => Ok(1),
    })?;
```

Token Transfer In CosmWasm, contracts are not called when tokens are sent to them, but they can query their current balance via the `deps.querier` parameter. Native tokens can be sent with a contract call. To access the amount sent, the contract can query the `info.funds` parameter. Transferring tokens from the contract to another address is done via a `BankMessage` attached to the response, following the actor model. See Listing 1.3 for an example of the latter.

Listing 1.3. A token transfer using the Cosmwasm SDK

```
let bank_msg = BankMsg::Send {
    to_address: caller.to_string(),
    amount: vec![coin.to_owned()],
};
let resp = Response::new().add_message(bank_msg);
Ok(resp)
```

Contract Metadata The contract can find its own address in the `Env` parameter. Token balance for any address can be queried by sending a `BankQuery` through `deps.querier`.

Blockchain State and **Notion of Time** Information about the current state of the blockchain is available in the `Env` parameter. This information includes block height and the timestamp of the block. Cosmwasm offers also an API, `deps.api`, to check the validity of an address.

Call Smart Contracts Calling another contract or module is done by adding a new `ExecuteMessage` to the response of the current message. See Listing 1.4 for an example. To act on the result of the call, the message is encapsulated in a submessage with the identifier of the current function and added to the response. The dispatcher will send the result back to the `reply` entry point on the original contract, where the reply is dispatched to the correct code based on the identifier that was provided in the submessage.

Listing 1.4. Calling another contract using the Cosmwasm SDK

```
let erc_transfer_msg = erc721::ExecuteMsg::TransferNft {
    recipient: env.contract.address.to_string(),
    token_id: nft_id,
```

```
};
let erc_transfer_msg = WasmMsg::Execute {
    contract_addr: nft_contract.into_string(),
    msg: to_json_binary(&erc_transfer_msg)?,
    funds: vec![],
};
let resp = Response::new().add_message(erc_transfer_msg);
Ok(resp)
```

Query Smart Contracts Querying other contracts or modules can be done via `deps.querier`. (See Listing 1.5)

Listing 1.5. Querying the state of another contract using the Cosmwasm SDK

```
let resp: erc721::OwnerOfResponse = deps.querier.query_wasm_smart(
            nft_contract_addr.to_owned(),
            &erc721::QueryMsg::OwnerOf { token_id: 1 },
    )?;
```

Events Events are attached to the response of the current message, similar to a `BankMsg` for a token transfer, and will then be published by the appchain.

5.2 Smart contracts in Polkadot

The ink! framework is an embedded domain-specific language (eDSL) to write smart contracts in Rust that compile to WebAssembly. Figure 4 offers an overview of the key interfaces using the ink! eDSL.

ink! offers its own CLI tool, `cargo-contract`, for setting up new contracts, compiling them, but also for interacting with the application-specific chain (called "parachain" in Polkadot).

The ink! eDSL uses Rust macros intensively to hide the complex underlying mechanics and make it more user-friendly. Each ink! smart contract is preceded by the `#[ink::contract]` macro and requires at least the following elements: exactly one `struct` with the `#[ink(storage)]` tag for storing the state of the

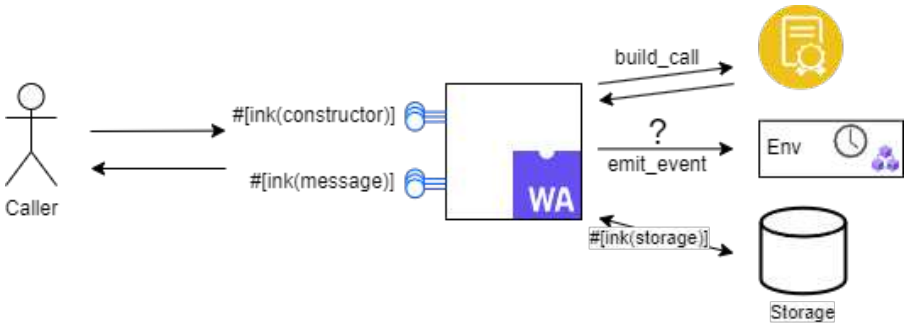


Fig. 4. Key interfaces of a smart contract on a Polkadot parachain using the ink! eDSL

contract, at least one function with the `#[ink(constructor)]` tag for initializing the contract on creation and at least one function with the `#[ink(message)]` tag which defines an entry point for the contract.

In Polkadot the bytecode of a contract is first uploaded to the blockchain. A new instance of the smart contract can then be created by initiating it with one of the defined constructors.

The messages that can be sent to the contract are defined as functions of the storage `struct`, which is a very common pattern in Rust. (See Listing 1.6)

Listing 1.6. Structure of an NFT smart contract implemented with the ink! eDSL

```
#[ink::contract]
mod erc721 {
    #[ink(storage)]
    #[derive(Default)]
    pub struct Erc721 {
        token_owner: Mapping<TokenId, AccountId>,
        ...
    }
    impl Erc721 {
        #[ink(constructor)]
        pub fn new() -> Self {...}
        #[ink(message)]
        pub fn balance_of(&self, owner: AccountId) -> u32 {...}
        #[ink(message)]
        pub fn transfer_from(&mut self, from: AccountId,
                             to: AccountId, id: TokenId,
        ) -> Result<(), Error> {...}
    }
}
```

Interaction with the blockchain environment is done via environment functions which are available via a public crate function `Self::env()` in the constructor or on `self.env()` for other methods.

ink! uses a synchronous approach to call or query other contracts which can be done in 2 different ways: using contract references or using a builder API.

To use contract references the contract to be called needs to be imported in the caller's project and the contract needs to be instantiated from that project. This can not be used to interact with contracts previously deployed on-chain.

The builder API can be used to call any smart contract compatible with the contracts palette. However, the API is low-level and as the interface of the other contract is not imported, it is not type-checked. See Listing 1.7 for an example.

Listing 1.7. Calling an external contract in ink! using the Builder API

```
let _my_return_value = build_call::<DefaultEnvironment>()
    .call(self.nft) //Contract address
    .call_v1()
    .gas_limit(0)
    .exec_input(
        ExecutionInput::new(Selector::new(ink::selector_bytes!(
```

```
7           "transfer_from")))
8               .push_arg(self.seller) //sender
9               .push_arg(self.env().account_id()) //address(this)
10              .push_arg(self.nft_id), //nftId
11      )
12      .returns::<ink::MessageResult<()>>()
13      .invoke();
```

We now analyse how the ink! eDSL covers the requirements from Section 4.

Message Endpoints A smart contract in ink! needs to have at least one public function with the `#[ink(constructor)]` macro for initializing the contract on creation and at least one public function with the `#[ink(message)]` macro to process state-changing calls and queries without side effects. Each functionality is implemented in a new public function.

Identification In most Polkadot parachains, user accounts and contracts are identified by a 256-bit `AccountId`, which is often the public key of a cryptographic key pair.

Ownership ink! has no built-in support for contract ownership. This means that contract ownership is handled in the same way as digital asset ownership, by storing ownership information in the contract's state. Native tokens are stored per `AccountId` and can be consulted via the `self.env().balance()` method.

Access Control Access control is managed by the contract itself with conditional statements and internally stored role information. The caller `AccountId` is retrieved via the `self.env().caller()` method.

Transactionality ink! uses a synchronous execution model. When the main contract fails, all state changes are rolled back, including contracts called by the main contract. In case the contract calls another contract, and that call fails, the main contract has the option to handle the failure and still commit its own state.

Persisting State Each contract has exactly one Rust `struct` with the `#[ink(storage)]` macro to store the contract state. ink! supports most common data types, provides its own version of `String` and `Vec`. It also allows `struct`s and `enum`s to be stored and provides a `Mapping` type. Initialization of the state is done in the constructors. Loading and saving is handled by the `#[ink(message)]` macro and is thus transparent for the developer. The macro will determine whether the method is allowed to change the state or not, in case of a query, by analyzing the method interface.

Token Transfer A method that can receive native tokens must be marked with the `#[ink(payable)]` macro. The value received with the call can be determined with `self.env().transferred_value()` and the total contract balance with `self.env().balance()`. Sending money from the contract is similar with `self.env().transfer(receiver_accountId, value)`.

Contract Metadata The contract `AccountId`, `balance` and the caller of the method are all made available through functions on `self.env()`.

Blockchain State Blockchain state can be queried through the `self.env()` functions, for example `self.env().block_number()`.

Notion of Time Like other Blockchain state information, the timestamp of the last block can be found with `self.env().block_timestamp()`.

Call Smart Contracts and **Query Smart Contracts** Both call and query are handled in a synchronous manner in ink!, depending on whether the contract is instantiated by the calling contract, it can be included in the project and allows for type checking at compile time. However if a call is to be made to a previously deployed contract, the usage of a low-level builder API is required.

Events Events are defined by a `struct` with the `#[ink(event)]` macro. And optionally the topic can be specified by `#[ink(topic)]` macro on the event fields, which let the parachain index the events based on the topic. Events are generated via the `self.env().emit_event(...)` function.

6 Qualitative and quantitative comparison of the SDKs

We now offer both a qualitative and a quantitative direct comparison between the Cosmwasm and Polkadot SDKs based on our first-hand experience in implementing the example contracts in each SDK. Table 3 summarizes the key differences between the SDKs.

Table 3. Key differences of Rust smart contracts in Cosmwasm and Polkadot SDKs.

Cosmwasm	**Polkadot**
Fixed set of entry points	Open-ended set of entry points
Asynchronous inter-contract calls (actor model). Type validation at compile time	Synchronous inter-contract calls. Prone to re-entrancy attack
Explicit key-value storage. Developer responsible for saving and loading of state	Implicit storage of a single annotated `struct`
Token transfer via `BankMessage` attached to the `Response`	`payable` attribute required on function to receive funds. Synchronous token transfer via `self.env().transfer()`
Contract ownership with `cw-ownable` crate	No built-in notion of contract ownership
Environment accessed via function parameters	Environment accessed via static functions
Events are attached to the `Response`	Events are emitted via a function call

Our quantitative comparison consists of measuring the total lines of code needed to implement both contracts in each SDK. Table 4 provides an overview of the count of non-comment, non-blank source lines of code (SLOC) of the smart contracts specified in Section 3 for Ethereum (Solidity), Cosmwasm (Rust) and Polkadot (Rust with ink! eDSL).[3] We took care to align the example code to have the same functionality, and we excluded any code related to unit tests in the count.

[3] Tokei, which supports both rust and solidity, was used to perform these measurements. `https://github.com/XAMPPRocky/tokei`

Table 4. Comparison of example smart contract implementation lines of code.

	Ethereum	**Cosmwasm**	**Polkadot**
NFT contract	72	313	184
Auction contract	63	318	217
Total	135	631	398

Perhaps unsurprisingly, Ethereum's Solidity language enables the most compact implementation with its tailored design. Both Cosmwasm and Polkadot Rust implementations are significantly larger. Part of this is due to the coding style and formatting rules of the Rust language. In addition, for Polkadot, the macro annotations add verbosity, as well as inter-contract calls with the builder API. Cosmwasm clearly has the most verbose implementation. This is in large part due to its actor model interface, with more lines needed to define custom message types and dispatching logic. However, the small difference in code count between the two example contracts indicates that once the setup is in place, adding extra functionality has less impact on the code count.

A final aspect of our qualitative comparison involves sharing some personal experiences gained from porting the Solidity code to both SDKs:

Documentation Both Cosmwasm and Polkadot offer tutorials and documentation, but generally the right information is scattered across many sources. The ink! documentation is more comprehensive, but still not complete enough to serve as a single reference. ink!'s extensive usage of Rust macros hides a lot of the complexity and results in a more straightforward migration from Solidity.

Scaffolding Polkadot offers a sample project to create a new smart contract. In Cosmwasm, a new smart contract project is a normal Rust library project, which needs to be manually customized or a template can be downloaded from `https://github.com/CosmWasm/cw-template`. We felt that a Cosmwasm project is more elaborate to set up, but once in place it is easy to extend and avoids accidental exposure of unwanted functionality, as each message type needs to be explicitly defined.

Testing Both SDKs offer frameworks that simulate the appchain and allow for end-to-end testing. We found the Polkadot toolchain to be easier to use. It offers a development node with initialised test accounts and a UI to facilitate contract interactions. The Cosmwasm development node requires more configuration and interaction can happen only via the CLI.

7 Related Work

Other comparative studies of smart contract languages and platforms exist in the literature, but they do not focus specifically on Rust or the SDK aspects of blockchain Dapp development. We review the most relevant studies below.

Bartoletti *et al.* [3] compare a variety of different smart contract languages, but do not focus on the use of Rust and Webassembly as the execution layer. The

authors do take into account the security implications of the language design and the tooling ecosystem, which is an aspect we want to consider for Rust SDKs as part of our future work (Section 8). Voloder and di Angelo [13] perform a comparative study of different smart contract platforms, but focus on the entire platform instead of the language and SDK aspects. Benahmed *et al.* [4] perform a comparative analysis from the development and performance perspective of the included platforms and only touch lightly on smart contract language design.

8 Conclusion and Future Work

We performed a comparative study of the Rust SDKs for developing smart contracts on the two most popular appchain platforms, Cosmwasm and Polkadot. Our study was guided by 13 requirements gathered from a representative pair of Solidity contracts, namely an English auction contract that can autonomously auction off an NFT (Non-fungible Token) managed by an NFT contract. Translating the two contracts to Rust using the Cosmwasm and Polkadot SDKs provides a baseline comparison on how the SDKs cover these requirements.

Our analysis creates a starting point for better understanding the design space of smart contract SDKs. Cosmwasm and Polkadot clearly occupy opposite points in this space, with Cosmwasm offering a library-based API, programmer-controlled storage and asynchronous inter-contract calls versus Polkadot's eDSL API, transparent storage and synchronous calls.

In future work we want to extend this research along three lines. First, we want to gain a deeper understanding of the impact of the SDK design on smart contract correctness (avoiding bugs), modularity (adding or changing functionality) and performance. Second, we want to extend the comparison to include additional use case scenarios (contracts) and other blockchain SDKs with support for WebAssembly (candidates include NEAR [11], Internet Computer [7] and Solana [15]). We also want to add a more in-depth section on the usability of the different SDKs and their ecosystems (user friendliness, available tooling, documentation quality, community support). Third, inspired by research into vulnerabilities in WebAssembly-based contracts [6] we want to extend the comparison from the Rust source code level to the WebAssembly bytecode level to understand how the SDKs expose blockchain host functionality to their guest Wasm contract modules.

Acknowledgments. This research is partially funded by the Research Fund KU Leuven, and by the Cybersecurity Research Program Flanders.

References

1. Agha, G.: Actors: a model of concurrent computation in distributed systems. MIT Press (1986)
2. Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on ethereum smart contracts (sok). In: POST 2017. Springer (2017)

3. Bartoletti, M., Benetollo, L., Bugliesi, M., Crafa, S., Sasso, G.D., Pettinau, R., Pinna, A., Piras, M., Rossi, S., Salis, S., Spanò, A., Tkachenko, V., Tonelli, R., Zunino, R.: Smart Contract Languages: A comparative analysis (Apr 2024)
4. Benahmed, S., Pidikseev, I., Hussain, R., Lee, J., Kazmi, S.A., Oracevic, A., Hussain, F.: A Comparative Analysis of Distributed Ledger Technologies for Smart Contract Development. In: PIMRC. IEEE (Sep 2019)
5. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform. (2014), https://ethereum.org/en/whitepaper/
6. Chen, W., Sun, Z., Wang, H., Luo, X., Cai, H., Wu, L.: Wasai: uncovering vulnerabilities in wasm smart contracts. In: ISSTA 2022 (2022)
7. DFINITY Team: The internet computer for geeks. Cryptology ePrint Archive, Paper 2022/087 (2022), https://eprint.iacr.org/2022/087
8. Erc-721: Non-fungible token standard (2018), https://eips.ethereum.org/EIPS/eip-721
9. Haas, A., Rossberg, A., Schuff, D.L., Titzer, B.L., Holman, M., Gohman, D., Wagner, L., Zakai, A., Bastien, J.: Bringing the web up to speed with webassembly. In: PLDI 2017 (2017)
10. Kwon, J., Buchman, E.: Cosmos: A network of distributed ledgers (2016), https://cosmos.network/whitepaper
11. NEAR Foundation: The official near white paper (2021), https://pages.near.org/papers/the-official-near-white-paper/
12. Solidity by example (Jun 2024), https://solidity-by-example.org/
13. Voloder, A., Di Angelo, M.: Comparison of Smart Contract Platforms from the Perspective of Developers. In: Wang, Q., Feng, J., Zhang, L.J. (eds.) Blockchain – ICBC 2023. Springer (2023)
14. Wood, G.: Polkadot: Vision for a heterogeneous multi-chain framework (2016), https://polkadot.network/whitepaper
15. Yakovenko, A.: Solana: A new architecture for a high performance blockchain. Whitepaper (2018)

Offchain Runtime Verification (for The Tezos Blockchain)*

Margarita Capretto[1,2], Martin Ceresa[1], Felipe Gorostiaga[1,3], Fernando Macias[1], Paloma Pedregal[1], and Cesar Sanchez[1]

[1] IMDEA Software Institute, Pozuelo de Alarcón s/n, 28223 Madrid, Madrid, Spain
[2] UPM, Madrid, Spain
[3] CIFASIS,Argentina

Abstract. In this paper, we present *offchain runtime verification*, a dynamic analysis technique to inspect blockchain executions without affecting the blockchain itself.
Runtime verification (RV) is a technique that analyzes traces of system execution based on monitors created from system specifications. There are two flavors of RV: online and offline. In online RV, monitors run in tandem with the system, either with their own resources or as code inlined in the system implementation. In offline RV, monitors have a dump of the system trace available. Examples of offline monitoring include post-mortem analysis and log inspection.
We present a novel notion of monitors running offchain while fetching information about the blockchain evolution and its agents (e.g. external users, bakers) to assess security and fairness, assign blame, and compute explanations. Our monitoring infrastructure is both *online*—as the monitors can receive new blocks incrementally—and *offline* since the monitors can query the history of the blockchain. Online queries are necessary because monitors are created after the blockchain has been running and relevant information is discovered online (e.g. who interacted in the past with an address recently discovered to be malicious). We describe in this paper an RV infrastructure for offchain monitoring for the Tezos Blockchain.

1 Introduction

Blockchains [30] running smart contracts [43, 45] provide a trusted third party where transactions are persistent and permanent. Smart contracts are immutable pieces of code (the code is the contract) that govern the interaction between agents using a blockchain without requiring a trusted centralized authority. We can use smart contracts to describe sophisticated functionality, enabling many

* This work was funded in part by PRODIGY Project (TED2021-132464B-I00)—funded by MCIN/AEI/10.13039/501100011033/ and the European Union NextGenerationEU/PRTR—by DECO Project (PID2022-138072OB-I00)—funded by MCIN/AEI/10.13039/501100011033 and by the ESF—and by a research grant from Nomadic Labs and the Tezos Foundation.

applications like decentralized finances (DeFi), decentralized governance, and Web3. Smart contracts are deterministic, i.e. the effects of executing smart contracts are uniquely determined by the blockchain state and the transactions parameters. Since smart contracts are immutable and they govern the blockchain evolution (including the cryptocurrency exchanged), the correctness of smart contracts is crucial and errors and vulnerabilities can lead to huge losses (e.g. [37]). Both static [1, 6, 7, 11, 31, 36, 42]and dynamic techniques [2, 8, 14, 26] have been proposed to approach the problem of smart contract correctness. Dynamic techniques analyze the evolution of the blockchain. Specifications describe correctness criteria for smart contracts and monitoring code is generated which extends the code of the contract. At runtime, monitors inspect smart contract invocations, detecting violations and reverting illegal executions. This *onchain* monitoring approach requires monitors to be deployed on-chain as part of smart contracts themselves, because otherwise, once smart contracts commit their effects cannot be reverted. Onchain monitoring, in turn, affects the normal execution of smart contracts as monitors consume some gas.

In this paper, we explore an alternative monitoring technique where monitors are deployed in a running system. Additionally, we seek non-intrusive monitors, so that the execution of smart contracts is completely unaffected by the execution of monitors. In these scenarios, monitors only observe a suffix of the system original trace. There are three possible approaches to cope with this lack of past observability: (1) *ignore the missing past* so monitors operate as if they were observing the whole history, which is the simplest approach but can lead to inaccurate results; (2) *encode the lack of knowledge* by modifying the specification [22]; (3) *access a log system* to fetch the missing past and then continue monitoring online with future events. We propose in this paper to follow the third approach by combining *offline* and *online* monitoring.

Runtime verification (RV) is a formal method for analyzing execution traces, one at a time. Traces are evaluated against monitors built from a given formal specification [4, 25]. Formal specifications are described using different languages implementing different logics like linear temporal logic [39]. Most RV languages describe a monolithic monitor that processes input events. Another approach is dynamic parametrization, also known as parametric trace slicing, which quantifies over objects and spawns monitors that follow independently the objects observed as in Quantified Event Automata (QEA) [3]. One can think of it as grabbing a magnifying glass when required.

Our approach is based on Stream Runtime Verification (SRV), pioneered by Lola [13], which relates output streams of verdicts to input streams of observations. Originally designed for testing synchronous hardware, SRV has since extended to other applications, including asynchronous and real-time systems (e.g. [18]). HLola [9, 19, 21] is an implementation of Lola as an embedded DSL in Haskell, which simplifies the specification development and runtime system. HLola leverages Haskell data types for Lola streams. Our main technical contribution is the extension of HLola with functionalities for *retroactive dynamic parametrization*, where parametrized specifications can be specialized with infor-

mation discovered dynamically (parametrization) and specifications can revisit past events to obtain missing information (retroactivity).

The inspection of the blockchain evolution is a perfect application of retroactive dynamic parametrization. The blockchain state, that is, the state of each smart contract and wallet, is connected to the next state and monitors can observe the whole execution history of the system. The challenge is to design efficient monitoring runtime systems that compute only what is necessary to produce a verdict.

Previous works have already inspected the blockchain evolution, mostly for security concerns within blockchain ecosystems [5,10,12,16,23,24,27,28,28,29,33, 34,38,40,41,44]In general, these works focus on specific problems and are tailored for performance but not correctness. In contrast, our framework is designed to be applied to several scenarios, and it allows writting diverse specifications ensuring that the monitors generated are correct with regard to the specification.

Running Example: Sandwich attacks. We introduce our running example, a blockchain vulnerability known as sandwich attack. Decentralized Exchanges (exchanges from now on) are a common application ofy blockchains to decentralized finances (DeFi) where users trade tokens directly without intermediaries. Sandwich attacks exploit the delay between transaction submission and transaction execution. When transactions are submitted for execution in blockchains they are first added as requests to a distributed service, known as mempools, containing transaction requests that have not yet been executed. The mempool content is visible to all agents in the blockchain, including malicious actors which inspect the mempool searching for victim transactions. In a sandwich attack, a victim tries to trade a large amount of token `A` for `B` in an exchange which will cause an increase in the price of token `B`. A malicious actor tries to "sandwich" the victim transaction with two additional transactions to profit from the future value increase of `B`. The first transaction, known as the *frontrunning* transaction, buys tokens `B` and is scheduled before the victim transaction. In the second transaction, known as the *backrunning* transaction, the malicious actor sells the purchased tokens `B` at a major price obtaining a profit.

The frontrunning transaction increases the price of token `B` causing the victim transaction to purchase `B` at a higher price than expected. Since token prices often fluctuate, the price of a given token when a transaction is submitted might differ from the actual transaction execution. Hence, most exchanges offer a way to define price ranges where token purchases can occur, which limits the amount of tokens `B` that the malicious actor can buy in the frontrunning transaction. This limits malicious actor's profits, but it does not prevent sandwich attacks from happening.

Consider a user trading a large amount of Tezos (XTZ) for USDT on an exchange. A malicious account can perform a sandwich attack purchasing USDT with XTZ in the same exchange right before the victim's transaction, and then selling USDT for XTZ right after.

In the remainder of the paper, we focus on detecting sandwich attacks against a specific account, denoted by `a`. We say that an account is *malicious* if it per-

forms a sandwich attack to account `a`. We also identify *suspicious* wallets as those that interacted directly with malicious accounts.

We implemented retroactive dynamic parametrization in HLola and report the result of applying our implementation to detect sandwich attacks in the Tezos blockchain [17]. An early prototype of this technique [35] was already used to efficiently detect distributed denial of service attacks in realistic network traffic. The contributions of this paper are:

- A new monitoring technique and its application for inspecting blockchain histories, described in Section 3.
- A demonstration of how to apply the features of our framework to detect sandwich attacks, identify involved actors, and compute attackers' profits and victims' losses, shown in Section 4.
- Further applications of our monitoring framework, presented in Section 5.

2 Preliminaries

We introduce now necessary concepts of Blockchains and SRV.

Blockchains. Blockchains [30] were introduced as distributed infrastructures that eliminate the need of trust third parties in electronic payment systems. Modern blockchains incorporate smart contracts [43, 45], stateful programs stored in blockchains controlling the functionality of blockchain transactions. Users interact with blockchains by invoking smart contracts. Blockchain "actors" (users and smart contracts) are identified by their *account*. We refer to accounts managed by end-users as *wallets*.

A *node* is a machine that stores a copy of the blockchain (or at least a portion of it) and keeps its local copy updated by regularly communicating with other nodes in a peer-to-peer network. Public blockchains allow anyone to launch a fully functional node. While nodes hold the entire history of the blockchain, searching this data directly can be slow and resource-intensive. Therefore, there is an ecosystem of tools, called *indexers*, that retrieve information from nodes and process it to allow efficient search. Indexers crawl the whole blockchain and store its data plus some additional information about the evolution of the blockchain, offering an API to query this information. Each API restricts the vision of the blockchain to what can be retrieved by such API language.

Stream Runtime Verification. Stream Runtime Verification (SRV) enriches monitoring algorithms from runtime verification to handle arbitrary data. SRV separates the logic of how data relates over time from the specific operations of each datatype. In this paper, we use the extensible tool HLola [9, 19, 21], an implementation of Lola [13] developed as an embedded DSL in Haskell.

Lola specifications consist of a set of typed input and output streams that represent the input events observed by the monitor and the intermediate observations and outputs of the monitor, respectively. Specifications are defined as

equations that declaratively describe the intended values of every output stream variable in terms of the input and output streams. The set of *stream expressions* of a given type is built from constants and function symbols as constructors, and from *offset expressions* of the form s`[`k`|`d`]` where s is a stream variable, k is an integer number and d is a default value of the type of s.

For example, offset expression `balanceA[-1|0]` represents the value of stream `balanceA` in the previous step of time with `0` as the value used at the initial instant. We define a stream `balA_ok` which checks that the `balance` of account A is always above a predefined threshold of `100` tokens:

```
input Int balanceA
output Bool balA_ok = balanceA[now] > 100
```

Given values of the input streams, the formal semantics of a Lola specification is defined denotationally as the unique collection of streams of values satisfying all equations.

One of the benefits of the extensible tool HLola is its ability to define templates for stream definitions using *static parametrization*. These templates act as abstractions, hiding specific concrete values, which are instantiated in static time by the compiler. Following the previous example, we can define a more generic version of the stream `balA_ok` as follows:

```
input Int balanceA
output Bool balA_checker <Int threshold> = balanceA[now] > threshold
output Bool balA_ok = (balA_checker 100)[now]
```

However, static parametrization cannot handle parameters whose values are discovered at runtime. The values of all parameters must be determined before the monitor starts executing. Users must ensure that the resulting specification contains a finite number of streams.

3 System Architecture

Our solution is composed of a monitor generated from an HLola specification and an external component interfacing with Tezos nodes and indexers called *adapter*. When monitors start execution, we start an adapter process in charge of receiving data from the Tezos blockchain and formatting it for the monitor input. Once the monitor is online, up and running, it can send parametrized queries to the adapter to fetch subtraces of the blockchain history. The adapter can perform complex requests to the Tezos indexer, i.e. filtering and formatting the received data before redirecting the result to the monitor. Through the use of the adapter, monitors efficiently obtain newly relevant data that was previously omitted or they can process blocks that were added to the blockchain before the monitor was launched. The adapter allows monitors to be agnostic to the

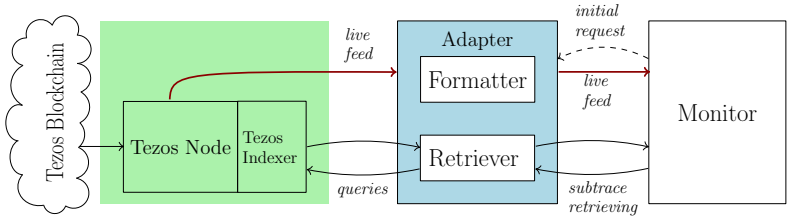


Fig. 1. Offchain monitoring system architecture

blockchain used, the indexer and the format of the data, so this architecture can be used (adapting the adapter) to other blockchains like Ethereum. Fig. 1 shows this architecture.

4 Features

In this section, we present the features of our framework and demonstrate how to apply them for effective monitoring, including to detect sandwich attacks, identify malicious and suspicious actors, and calculate attackers' profits.

4.1 Monitor Side Features

Nested monitors. Nested monitors [20] allow using SRV specifications as data functions inside other specifications. Nested specifications are created and executed dynamically. A nested monitor receives a finite subtrace of the system original trace as input, typically obtained using HLola operator s`[:`n`]`, which creates a list with the next n values of stream s.

To define a *nested* specification, we need to provide a name so we can refer to it later on and add an extra clause: `return` x `when` y where x is a stream of any type and y is a `Boolean` stream. The type of the stream x determines the type of the value returned when the specification is invoked dynamically. The Boolean stream y dictates when the nested specification finishes. The nested monitor returns the value of x at the first instant at which y becomes **true** (without needing to inspect the rest of the list), or the last value of x if y is never **true**. Nested specifications can be parametric, parameters are declared after its name. We can execute nested specifications by using the HLola function `runSpec` specifying the parameters and the input streams with which the nested monitor will be executed.

Example 1. The following specification calculates whether a transaction in the input stream `tx` is the frontrunning transaction of a sandwich attack against account `a`.

Blockchain traces may contain many trading operations involving the same pair of tokens. In a sandwich attack, the frontrunning and backrunning transactions must occur close in time to the victim transaction. For simplicity, we

consider an operation to be part of a potential sandwich attack if both the suspected frontrunning and backrunning transactions occur within 10 transactions apart from the victim transaction. We define a stream `frontruns` which first checks if tokens are traded in the transaction using function `tradeTokens`. If so, the monitor invokes the nested specification `frontrunspec` with the current transaction (`tx[now]`) as the parameter `ftx`, and the next 20 events of `tx` (`tx[:20]`) as the input stream `tx` of the nested monitor.

```
use innerspec frontrunspec
input Transaction tx
output Bool frontruns = if tradeTokens tx[now]
                        then runSpec (frontrunspec tx[now] tx[:20])
                        else False
```

We use `if` · `then` · `else` · instead of the Boolean operator `(&&)` to stress the fact that the nested specification is only executed when tokens are traded. The nested specification `frontrunspec` defines three streams.

- Stream `counter` counts the number of transactions processed. We use it to guarantee that the victim transaction is among the first 10 transactions, and that the backrunning transaction occurs within 10 transactions of the victim transaction.
- Stream `victim` stores the position in the input trace of a potential victim transaction (a transaction where account `a` trades tokens in the same exchange as the frontrunning transaction).
- Stream `attack` indicates if an attack occurred (the client from the frontrunning transaction swapped tokens back at most 10 transactions after the victim transaction).

```
innerspec Bool frontrunspec <Transaction ftx>
input Transaction tx
const a = "tz123"
output Int counter = counter[-1|0] + 1
output Int victim =
  if counter[now] < 11 &&  tradeTokens tx[now]
     && exchange tx[now] == exchange ftx && client tx[now] == a
     && token1 tx[now] == token1 ftx && token2 tx[now] == token2 ftx
  then counter[now]
  else -1
output Bool attack =
  victim[now] != -1 && counter[now] < victim[now] + 11
  && tradeTokens tx[now] && exchange tx[now] == exchange ftx
  && client tx[now] == client ftx
  && token1 tx[now] == token2 ftx && token2 tx[now] == token1 ftx
return attack when attack
```

In a transaction where tokens are traded, the `client` of a transaction is the account that traded the tokens, the `exchange` of a transaction is the exchange where the trade happened, and `token1` and `token2` of a transaction are the traded tokens. We can further extend the above specification with a stream called `malicious` to track accounts that performed sandwich attacks against `a`.

```
output (Set Account) malicious = if frontruns[now]
  then insert (client tx[now]) malicious[-1|empty]
  else malicious[-1|empty]
```

Retroactive Nested monitors. Although nested monitors can detect sandwich attacks, this approach can be very inefficient. In a blockchain history, the number of times account `a` trades tokens (and can be victim of a sandwich attack) is significantly smaller than the total number of transactions where tokens are traded. This leads to a large number of unnecessary nested monitors being created. To address this inefficiency, we propose creating nested monitors only when account `a` trades tokens. This implies ignoring relevant transactions and later accessing them to search for potential frontrunning transactions. We achieve this *retroactive* search by implementing a function `pastRetriever` that invokes the adapter (see Section 3) to retrieve a specified number of past events.

The following specification checks whether the current transaction corresponds to address `a` trading tokens, and triggers the finer analysis of the surrounding transactions when necessary.

```
use innerspec tradersSpec
input Transaction tx
const a = "tz123"
output Bool attacked =
  if tradeTokens tx[now] && client tx[now] == a then
    let fRunners = runSpec ((tradersSpec e t1 t2) (pastRetriever 10))
        bRunners = runSpec ((tradersSpec e t2 t1) tx[:10]) in
        not (null (intersection fRunners bRunners))
  else False
  where e = exchange tx[now]
        t1 = token1 tx[now]
        t2 = token2 tx[now]
```

Within the 10 previous transactions, nested specification `tradersSpec` computes all accounts that made the same trade as `a`. On the subsequent 10 transactions, nested specification `tradersSpec` computes all accounts that did the opposite trade. Finally, the specification checks whether any account appears in both sets. We use `if · then · else False (&&)` to stress the fact that the nested specification is executed only when `a` trades tokens.

The nested specification `tradersSpec` identifies all accounts that trade two specific tokens in a given exchange, as follows:

```
innerspec (Set Account) tradersSpec <Account e> <Token t1> <Token t2>
input Transaction tx
output (Set Account) traders =
  if tradeTokens tx[now] && exchange tx[now] == e
     && token1 tx[now] == t1 && token2 tx[now] == t2
  then insert (client tx[now]) traders[-1|emptySet]
  else traders[-1|emptySet]
return traders when False
```

The above specification finds all transactions that are victims of a sandwich attack at most 10 transactions *after* it happens. Also, the nested monitors in this example are created, executed and destroyed only for every transaction where `a` exchanges tokens.

(Forward) Dynamic Parametrization. Since we have an efficient way of detecting sandwich attacks and malicious accounts, we can move on to identifying suspicious wallets, defined as those that interact with the malicious account performing the sandwich attack. The following specification computes all wallets that interact with a given account:

```
input Transaction tx
output (Set Wallet) fellows <Account a> =
  union (wallets a tx[now]) fellows[-1|empty]
```

The auxiliary function `wallets a tx` returns all wallets that sent tokens to account `a` during the execution of transaction `tx`. To identify all suspicious wallets, we need to instantiate the parametrized stream `fellows` with all malicious accounts. However, malicious accounts are only found after they perform a sandwich attack, which cannot be determined statically.

We can instantiate a parametric stream over values discovered dynamically while processing the input trace using the HLola operator `over`. The `over` operator takes two arguments: (1) a parametric stream `strm`, and (2) a stream `params` of sets of values. The resulting expression is a map where at any point in time the keys are the elements in `params[now]`, and the value associated to each key is the instantiation of `strm` over the key. For a complete description on how this operator is implemented in the tool HLola, see [35]. In our case, we can parametrize the parametric stream `fellows` over the values of stream `malicious`:

```
output (Set Wallets) suspicious = foldl union empty
  (elems (fellows `over` malicious))
```

Here, `elems m` returns the list of values in map `m`, and function `foldl f l` aggregates the elements in list `l` using `f` to combine them. To compute all suspicious

wallets, we join the suspicious wallets related to each malicious account. This specification follows each account independently.

When a new value is added to the set of parameters (the stream `malicious` in our case), we spawn a new monitor parametrized with the discovered value. The newly created nested monitor executes alongside the monitor that created it, as long as its associated parameters remain part of the set represented by stream `malicious`.

The nested monitors used for forward dynamic parametrization process the same events as the root monitor, but it is often the case that only some of the events are relevant to a specific parametrized stream. We can use subtracing to redefine stream `suspicious` as follows:

```
output (Set Wallets) suspicious = foldl union empty
  (elems (fellows `over` malicious `updating` (accounts tx[now])))
```

where `accounts` returns the set of all accounts involved in a transaction. The `updating` operator lets us specify the parameters of the monitors that have to process the current event.

In the example above, the monitor follows the dynamically parametrized stream once the parameter has been discovered (like in Lola2.0 [15] or in quantified event automata QEA [3]). However, monitoring a stream only after its parameter is discovered has its limitations, for example that the beginning prefix of the trace is ignored. In our example, this means that the monitor cannot discover wallets that interacted with a malicious account before the malicious account is identified, e.g. before a sandwich attacker reveals its identity.

We could still use *forward dynamic parametrization* to identify all wallets that ever interacted with malicious accounts, regardless of when the interaction happened. To achieve this, we need to follow all created accounts, tracked by stream `allaccounts`, and then, at every instant, keep only the wallets related to the malicious accounts discovered so far, using the stream `malicious`.

```
output (Set Account) allaccounts =
  union (createdAccounts tx[now]) allaccounts[-1|empty]
output (Set Wallet) suspicious = foldl union empty
  (elems (filterWithKey ismalicious
    (suspicious `over` allaccounts `updating` (accounts tx[now]))))
  where ismalicious k _ = member k malicious[now]
```

The function `filterWithKey p m` filters all the key-values in map `m` that satisfy the predicate `p`. Although this specification is correct, if most accounts are not malicious, this forward monitor follows many accounts unnecessarily.

However, as part of our infrastructure we have the node and indexer storing the past events of the trace, so we can combine *retroactive nested monitors* with *dynamic parametrization* when a new parameter is discovered, effectively implementing *retroactive dynamic parametrization*.

Retroactive Dynamic Parametrization. Retroactive dynamic parametrization [35] is a technique that allows monitors to revisit the past of the trace whenever a new parameter is discovered, initialize the parametric stream with the retrieved information and continue monitoring online from that point onward. This nested monitor behaves exactly as a forward parametrized monitor where an oracle had correctly guessed which parameters would be found to be useful and which parameters can be ignored. Retroactive dynamic parametrization is implemented by adding a new clause `withInit` to the `over` operator. This clause allows specifying an *initializer*, which initializes the nested monitor with events taken from the trace up to the current point. Typically, an initializer involves calling an external program (see Section 3) that interacts with an offline infrastructure to efficiently retrieve relevant past trace elements based on the discovered parameter.

We can use *retroactive* monitoring to only create the dynamic parameters when the corresponding account is malicious, and use the retroactive capability to inspect the past of the trace and see which wallets interacted with them in the past. We redefine the stream `suspicious` accordingly.

```
output (Set Wallets) suspicious = foldl union empty
 (elems (suspicious `over` malicious `updating` accs `withInit` initer))
 where accs = accounts tx[now]
```

The new `over` expression specifies an initializer `initer` (whose definition is not shown in the specification) that calls the adapter to retrieve the past of the corresponding parameter. The adapter uses the indexer to efficiently retrieve only the events in the past relevant to the current account.

4.2 Execution Simulation

To further analyze sandwich attacks, one could be interested in determining the profit obtained by the attacker. This requires reasoning about what would have happened if the invocations to the blockchain had been different. In our example, we are interested in comparing what did happen (in particular the legitimate exchange invocation) with what would have happened if the attack had not existed. For these questions, our monitoring infrastructure introduces a simple *simulation* framework.

The blockchain state is public and the code of smart contracts code is available, and evaluation frameworks are typically provided by the blockchain developers (using the exact same code that bakers execute). We use the official Tezos interpreter in our monitoring infrastructure to perform a small-step execution machine of alternative executions and observe the blockchain intermediate states. We have developed two basic building blocks:

– **Data crawler:** for a given set of operations (blocks or groups) the data crawler queries the blockchain extracting which contracts were involved in the set of transactions requested.

– **Simulation:** given a set of contracts and their state, execute a sequence of grouped transactions in order.

This allows monitors to simulate operations that happened in the blockchain and also to explore alternative histories. To simulate operations that happened, we first get the required information to execute the operations, that is, the invoking smart contract and its state, plus all other invoked smart contracts and their states. Since we know what happened because it is publicly available on the blockchain node, we can determine the smart contracts involved in a given transaction. Once we have the initial states of every contract involved, we can just execute one transaction at a time replicating the behavior executed by the blockchain.

If we diverge from transactions that happened, as it would happen if we are executing hypothetical scenarios, we may get into missing some contract states. To explore alternative histories, we first obtained the contracts that were called during a possible execution. Then, we perform a hypothetical execution, which may lead to the invocation of smart contracts whose storages were not fetched. We detect the address of the missing contract, add it to the list of required addresses for execution using the data crawler and iterate until we finish execution.

We can extend our running example about detecting sandwich attacks computing the damage suffered by the victim simulating an alternative execution in which the sandwich attack does not exist and comparing the hypothetical and the real balance of the victim.

```
input Transaction tx
input Double balanceA
const a = "tz123"
output Double stolen =
  if attacked[now]
  then (getBalance a (simulate txs')) - balanceA[now]
  else 0
  where txs' = filter (not frontruns) (pastRetriever 10)
        frontruns t = client t /= a && exchange t == exchange tx[now]
        && tokens1 t == tokens1 tx[now] && tokens2 t == tokens2 tx[now]
```

In the example above, we only expose the difference between the real and alternative balances of the victim. To precisely quantify how much the attacker stole, the monitor can also inspect manually the valuable items (as tokens) and describe the computation as an arithmetic expression. In summary, previous specifications detect and extract transactions causing sandwich attacks, filter them and observe the state of the blockchain as if these transactions had not been executed.

Number of monitored blocks	50,000
Number of transactions in monitored blocks	2,624,594
Number of transactions since the beginning of the blockchain	288,705,340
Number of calls to 3route v4	2,278
Number of attacks	39
Number of malicious accounts	3
Number of suspicious accounts	97
Number of suspicious wallets	5

Fig. 2. Summary of sandwich attack monitoring.

5 Case Studies

1. Detecting Sandwich Attacks. We used our framework to implement a retroactive dynamic monitor for the detection of sandwich attacks, identification of malicious addresses and quantification of the losses incurred.

The monitor receives a stream of transactions from the Tezos blockchain and searches for possible sandwich attacks. In this case study, we set the victim account `a` to an exchange aggregator smart contract known as *3route v4*.

We executed the monitor starting from block 5,200,000 in the Tezos main net until block 5,250,000. Retroactive parametrization allows us to start the monitor at any point in the blockchain and find fund transfers that happened before the monitor was launched. The table in Fig. 2 sums up the results obtained. Thanks to retroactive monitoring, we obtained the suspect accounts without analyzing every transaction. Instead of following 288 million transactions, the monitor only queries the adapter for the past transactions when a specific target is dynamically obtained, in the infrequent event where a suspicious account is found. Furthermore, the search for the frontrunning and backrunning transactions is only performed when the monitor detects a call to the exchange aggregator, which occurred only in 0,08% of the monitored transactions.

2. Clustering. In this case study, we consider that an address that performs front-running is a *malicious* address, and we mark the addresses that transferred cryptocurrency in the past to a malicious address as the potential source of funds is a *suspicious* address. We leave out of the search addresses that transferred funds to suspicious accounts.

When we start to follow indirectly related addresses, we find that they form *clusters* of heavily-interacting accounts with prominent addresses that act as interconnecting hubs between clusters. The flexibility of HLola allowed us to develop several implementations of clustering algorithms to discover the degree of suspiciousness of a wallet with respect to a malicious account based on how many times they interact (directly or indirectly), how many funds they exchange (directly or indirectly), and how many intermediaries are in their relation.

3. Juster. Juster is a decentralized application that allows Tezos users to bet on events that represent the changes of certain cryptocurrency prices within a given time interval. Users get a reward if their predictions are correct and lose their bet otherwise. For example, users can bet that the value of the Tezos cryptocurrency XTZ will rise by 10% or more in the following day. The Juster administrator opens events on which the users can bet and closes them after the betting interval ends, distributing the earnings accordingly.

We define an HLola monitor for the Juster platform assessing that:
(1) all closed events were previously open and no open event is reopened;
(2) there are less than 100 open events at any given time.
The monitor receives events tagged with an identifier `eventId` and with the kind of event which can be either `Open` or `Close`. We define the specification in HLola:

```
input EventId eventId
input Operation operation
define {EventId} open_events =
  if operation[now] == Open
    then insert(eventId[now], openevents[-1|{}])
  else if operation[now] == Close
    then delete(eventId[now], openevents[-1|{}])
  else openevents[-1|{}]
output Bool few_events = size(openevents[now]) < 100
output Bool right_order =
  (operation[now] == Close) == member(eventId[now], openevents[-1|{}])
```

4. BFS vs DFS in Tezos. Tezos is a self-amending blockchain that provides a mechanism to change its rules through regular protocol upgrades. Protocol Florence [32], modified the execution order of operations between smart contracts, switching from a breadth-first search (BFS) to a more conventional depth-first search (DFS) algorithm. This change in the execution order can potentially impact transactions outcomes. In this case study, we identified those transactions that could have behaved differently under the two execution orders.

A naive approach is to simulate each transaction under both execution orders and compare the results. However, this approach is very inefficient for the entire blockchain because simulating requires access to all invoked smart contracts and their states (see Section 4.2). Fortunately, most transactions are guaranteed to behave the same under BFS and DFS without simulation, because the execution order only affects if some smart contract is invoked twice and the order of the calls differs between execution orders. This is because the state of a contract only varies when the contract is invoked. The difference in the call order to a smart contract can be detected by inspecting the transaction call graph (a directed tree where nodes are labeled with smart contracts and edges represent calls).

Unfortunately, indexers do not store the transactions call graph, but only the call sequence. For each transaction, the monitor creates all possible call graphs that can generate the given call sequence when traversed with the corresponding execution order. If in one of the call graphs, the call order for a smart contract differs between BFS and DFS, the monitor marks that the transaction must be simulated.

For this case study, we considered all 34,856,986 transactions corresponding to the years 2021 and 2022. We used the adapter to retrieve from the indexer only those transactions in which some smart contract is called more than once, obtaining 1,260,145 transactions. For each transaction, then the adapter produces only its identifier, call sequence, and the execution order used when executing it. As the actual name and address of the smart contract invoked is irrelevant in this case, to save space, the formater assigns to each smart contract in a given transaction a unique small identifier. Finally, the monitor received all 1,260,145 transactions and detected that only 599,684 (out of 34,856,986) require simulation to determine behavioral differences under the other execution order.

6 Conclusions

We presented in this paper a framework for the offchain runtime verification of blockchains, and more specifically, for the Tezos Blockchain. Offchain monitoring allows us to create monitors which receive new blocks (as in online monitoring) and can perform retroactive queries to the past of the blockchain (as in offline monitoring). The retroactive feature is useful both for requesting information about the past, before the monitoring was created, and to lazily evaluate events that most of the time are irrelevant for the monitor.

We described our implementation based on stream runtime verification, and in particular on the HLola language, and several cases studies including the detection of sandwich attacks. Future work includes more advanced case studies and more quantitaitve evalutaion and comparison with other frameworks, which was beyond the scope of this work. Additionally, we plan to make the monitoring front-end available as a service, enabling its application for other blockchains.

References

1. Ahrendt, W., Bubel, R.: Functional verification of smart contracts via strong data integrity. In: Proc. of ISoLA. LNCS, vol. 12478, pp. 9–24. Springer (2020)
2. Azzopardi, S., Ellul, J., Pace, G.J.: Monitoring smart contracts: ContractLarva and open challenges beyond. In: Proc. of RV'18. LNCS, vol. 11237, pp. 113–137. Springer (2018)
3. Barringer, H., Falcone, Y., Havelund, K., Reger, G., Rydeheard, D.: Quantified event automata: Towards expressive and efficient runtime monitors. In: Proc of FM'12. LNCS, vol. 7436, pp. 68–84. Springer (2012)
4. Bartocci, E., Falcone, Y. (eds.): Lectures on Runtime Verification - Introductory and Advanced Topics, LNCS, vol. 10457. Springer (2018)

5. Bartoletti, M., Pes, B., Serusi, S.: Data mining for detecting bitcoin ponzi schemes. In: CVCBT'18. pp. 75–84 (2018)
6. Bernardo, B., Cauderlier, R., Hu, Z., Pesin, B., Tesson, J.: Mi-Cho-Coq, a framework for certifying Tezos smart contracts. arXiv **abs/1909.08671** (2019), http://arxiv.org/abs/1909.08671
7. Bhargavan, K., Delignat-Lavaud, A., Fourneta, C., Gollamudi, A., Gonthier, G., Kobeissi, N., Kulatova, N., Rastogi, A., Sibut-Pinote, T., Swamy, N., Béguelin, S.Z.: Formal verification of smart contracts: Short paper. In: Proc. of PLAS@CCS'16. pp. 91–96. ACM (2016)
8. Capretto, M., Ceresa, M., Sánchez, C.: Transaction monitoring of smart contracts. In: Proc of RV'22. LNCS, vol. 13498, pp. 162–180. Springer (2022)
9. Ceresa, M., Gorostiaga, F., Sánchez, C.: Declarative stream runtime verification (hLola). In: Proc. of APLAS'20. LNCS, vol. 12470, pp. 25–43. Springer (2020)
10. Christin, N.: Traveling the silk road: a measurement analysis of a large anonymous online marketplace. In: Proc. of ACM WWW '13. p. 213–224
11. Conchon, S., Korneva, A., Zaïdi, F.: Verifying smart contracts with Cubicle. In: Proc. of FMBC'19. LNCS, vol. 12232, pp. 312–324. Springer (2019)
12. Conti, M., Gangwal, A., Ruj, S.: On the economic significance of ransomware campaigns: A bitcoin transactions perspective. Computers & Security **79**, 162–189 (2018)
13. D'Angelo, B., Sankaranarayanan, S., Sánchez, C., Robinson, W., Finkbeiner, B., Sipma, H.B., Mehrotra, S., Manna, Z.: LOLA: Runtime monitoring of synchronous systems. In: Proc. of TIME'05. pp. 166–174. IEEE CS Press (2005)
14. Ellul, J., Pace, G.J.: Runtime verification of Ethereum smart contracts. In: Proc. of EDCC'18. pp. 158–163. IEEE Computer Society (2018)
15. Faymonville, P., Finkbeiner, B., Schledjewski, M., Schwenger, M., Stenger, M., Tentrup, L., Hazem, T.: StreamLAB: Stream-based monitoring of cyber-physical systems. In: Proc. of CAV'19. LNCS, vol. 11561, pp. 421–431. Springer (2019)
16. Gomez, G., Moreno-Sanchez, P., Caballero, J.: Watch your back: Identifying cybercrime financial relationships in bitcoin through back-and-forth exploration. In: Proc. of ACM CCS'22. p. 1291–1305 (2022)
17. Goodman, L.M.: Tezos – a self-amending crypto-ledger. https://www.tezos.com/whitepaper.pdf (2014)
18. Gorostiaga, F., Sánchez, C.: Striver: Stream runtime verification for real-time event-streams. In: Proc. of RV'18. LNCS, vol. 11237, pp. 282–298. Springer (2018)
19. Gorostiaga, F., Sánchez, C.: HLola: a very functional tool for extensible stream runtime verification. In: Proc. of TACAS'21. LNCS, vol. 12652, pp. 349–356. Springer (2021)
20. Gorostiaga, F., Sánchez, C.: Nested monitors: Monitors as expressions to build monitors. In: Proc. of RV'21. LNCS, vol. 12974, pp. 164–183. Springer (2021)
21. Gorostiaga, F., Sánchez, C.: Stream runtime verification of real-time event streams with the Striver language. STTT **23**, 157–183 (2021)
22. Gorostiaga, F., Sánchez, C.: Monitorability of expressive verdicts. In: Proc. of NFM'22. LNCS, vol. 13260, pp. 693–712. Springer (2022)
23. Huang, D.Y., Aliapoulios, M.M., Li, V.G., Invernizzi, L., Bursztein, E., McRoberts, K., Levin, J., Levchenko, K., Snoeren, A.C., McCoy, D.: Tracking ransomware end-to-end. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 618–631
24. Lee, S., Yoon, C., Kang, H., Kim, Y., Kim, Y., Han, D., Son, S., Shin, S.: Cybercriminal minds: An investigative study of cryptocurrency abuses in the dark web. Proc. of NDSS '19

25. Leucker, M., Schallhart, C.: A brief account of runtime verification. J. Logic Algebr. Progr. **78**(5), 293–303 (2009)
26. Li, A., Choi, J.A., an. Long: Securing smart contract with runtime validation. In: Proc. of ACM PLDI'20. pp. 438–453. ACM (2020)
27. Liao, K., Zhao, Z., Doupe, A., Ahn, G.J.: Behind closed doors: measurement and analysis of cryptolocker ransoms in bitcoin. In: 2016 APWG Symposium on Electronic Crime Research (eCrime). pp. 1–13
28. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. Commun. ACM **59**(4), 86–93 (2016)
29. Möser, M., Böhme, R., Breuker, D.: An inquiry into money laundering tools in the bitcoin ecosystem. In: 2013 APWG eCrime Researchers Summit. pp. 1–14
30. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), `https://bitcoin.org/bitcoin.pdf`
31. Nehaï, Z., Bobot, F.: Deductive proof of industrial smart contracts using Why3. In: Proc. of FMBC'19. LNCS, vol. 12232, pp. 299–311. Springer (2019)
32. Nomadic Labs: Protocol florence, `https://tezos.gitlab.io/protocols/009_florence.html`, Accessed: 2024-06-06
33. Paquet-Clouston, M., Haslhofer, B., Dupont, B.: Ransomware payments in the Bitcoin ecosystem. Journal of Cybersecurity **5**(1) (2019)
34. Paquet-Clouston, M., Romiti, M., Haslhofer, B., Charvat, T.: Spams meet cryptocurrencies: Sextortion in the bitcoin ecosystem. In: AFT '19. p. 76–88
35. Pedregal, P., Gorostiaga, F., Sánchez, C.: A stream runtime verification tool with nested and retroactive parametrization. In: Proc. of RV'23. LNCS, vol. 14245, pp. 351–362. Springer (2023)
36. Permenev, A., Dimitrov, D., Tsankov, P., Drachsler-Cohen, D., Vechev, M.: VerX: Safety verification of smart contracts. In: Proc. of S&P'20. pp. 1661–1677. IEEE (2020)
37. Phil, D.: Analysis of the dao exploit (2016), `https://hackingdistributed.com/2016/06/18/analysis-of-the-dao-exploit/`
38. Pletinckx, S., Trap, C., Doerr, C.: Malware coordination using the blockchain: An analysis of the cerber ransomware. In: CNS'18. pp. 1–9 (2018)
39. Pnueli, A.: The temporal logic of programs. In: Proce. of FOCS'77. pp. 46–67. IEEE CS Press (1977)
40. Ron, D., Shamir, A.: How did dread pirate roberts acquire and protect his bitcoin wealth? In: Financial Cryptography and Data Security. pp. 3–15. Springer Berlin Heidelberg (2014)
41. Spagnuolo, M., Maggi, F., Zanero, S.: Bitiodine: Extracting intelligence from the bitcoin network. In: Financial Cryptography and Data Security. pp. 457–468. Springer Berlin Heidelberg (2014)
42. Stephens, J., Ferles, K., Mariano, B., Lahiri, S., Dillig, I.: SmartPulse: Automated checking of temporal properties in smart contracts. In: Proc. of S&P'21. IEEE (2021)
43. Szabo, N.: Smart contracts: Building blocks for digital markets. Extropy **16** (1996)
44. Tekiner, E., Acar, A., Uluagac, A.S., Kirda, E., Selcuk, A.A.: Sok: Cryptojacking malware. In: IEEE EuroS&P '21. pp. 120–139
45. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper **151**, 1–32 (2014)

CBT Session 2: Consensus protocols & the cryptocurrency ecosystem

Quantifying Liveness and Safety of Avalanche's Snowball

Quentin Kniep[1], Maxime Laval[2], Jakub Sliwinski[1], and Roger Wattenhofer[1]

[1] ETH Zurich
Zurich, Switzerland
{qkniep,jsliwinski,wattenhofer}@ethz.ch
[2] EPFL
Lausanne, Switzerland
maxime.laval@epfl.ch

Abstract. This work examines the resilience properties of the Snowball and Avalanche protocols that underlie the popular Avalanche blockchain. We experimentally quantify the resilience of Snowball using a simulation implemented in Rust, where the adversary strategically rebalances the network to delay termination.
We show that in a network of n nodes of equal stake, the adversary is able to break liveness when controlling $\Omega(\sqrt{n})$ nodes. Specifically, for $n = 2000$, a simple adversary controlling 5.2% of stake can successfully attack liveness. When the adversary is given additional information about the state of the network (without any communication or other advantages), the stake needed for a successful attack is as little as 2.8%.
We show that the adversary can break safety in time exponentially dependent on their stake, and inversely linearly related to the size of the network, e.g. in 265 rounds in expectation when the adversary controls 25% of a network of 3000.
We conclude that Snowball and Avalanche are akin to Byzantine reliable broadcast protocols as opposed to consensus.

1 Introduction

The Avalanche protocol [13] advertises exceptional performance in terms of transaction throughput and latency. The Avalanche blockchain based on the protocol has certainly gained significant attention and support within the cryptocurrency community, as evidenced by the remarkable market capitalization of its native token amounting to \$10B[3]. The media prominence and monetary value firmly place Avalanche among the most popular and successful blockchain systems.

The protocol is built on a simple mechanism that operates by repeatedly sampling random nodes of the network in order to gauge the system's support of a given decision and confirm transactions. Conceptually, the underlying Snowball protocol can be compared to a voting process for a binary choice concerning a transaction. The protocol description promises to swiftly converge to a final

[3] https://coinmarketcap.com (Accessed: June 19 2024)

decision from a network state initially divided equally between two alternatives. With the aid of a directed acyclic graph (DAG), Avalanche forms a partial order of transactions instead of the total order that is established by usual blockchain protocols, like Bitcoin [12] or Ethereum [6]. Thus when transactions on Avalanche are accepted, validators are able to execute them in different orders based on their current view of the DAG, as long as those transactions are not causally dependent on each other. In theory, such structure can allow for a higher degree of parallelism in the transaction confirmation process, which can lead to a higher throughput than traditional blockchain protocols.

The ideas that form Avalanche stand in contrast to other Proof-of-Stake and BFT-based consensus protocols such as Ethereum 2.0. While the whitepaper [13] claims excellent resilience, it only proves the protocol's liveness in presence of up to $\mathcal{O}(\sqrt{n})$ malicious parties, where n represents the total number of validators (or stake supply). However, usually Proof-of-Stake protocols ensure the upper bound resilience of $\frac{n}{3}$ in partial synchrony.

Another detail that stands out in the description of Avalanche, is how it defines its guarantees with respect to "virtuous" transactions, i.e. assuming there's no conflicting alternative in the system. Remarkably, broadcast-based payment systems [9,4] are inherently reliant on such an assumption, and as such are fundamentally weaker than consensus protocols.

The lack of clarity about the Avalanche family of protocols begs the question: how resilient Snowball and Avalanche really are? Does the unusual consideration of "virtuous" transactions indicate a fundamental limitation?

Our Contribution We examine the resilience properties of the Snowball and Avalanche protocols.

We experimentally exhibit the resilience of Snowball against attacks from adversarial nodes. Our simulation showcases that in a system of n nodes (or stake supply), the adversary can indefinitely halt the Snowball protocol when controlling a stake of $\Omega(\sqrt{n})$, or less than 2% in some experimental scenarios. Furthermore, we examine a strategy for an adversary to violate safety by getting a single validator to finalize an output distinct from the rest of the network. The expected duration of the safety attack depends exponentially on the stake controlled by the adversary, and is inversely linear to the size of the network. For example, at 25% adversarial stake in a network of 3000, safety can be violated after 265 rounds in expectation.

We discuss how these considerations translate to the Avalanche protocol based on Snowball.

Finally, we draw parallels between Avalanche and broadcast-based payment systems, and conclude that Avalanche is fundamentally weaker than usual consensus protocols.

2 Background

Avalanche's blockchain platform consists of three distinct built-in blockchains: The Exchange Chain (X-Chain), the Contract Chain (C-Chain) and the Platform Chain (P-Chain) [14].

The **X-Chain** is responsible for processing simple transactions on the network, such as transfers of the native AVAX token. It is based on the Avalanche protocol with the DAG that runs multiple instances of the Snowball algorithm and only partially orders transactions.

The **C-Chain** is responsible for executing general smart contracts compatible with the Ethereum Virtual Machine (EVM). In contrast to the X-Chain, C-Chain uses the Snowman protocol which ensures a total order of all transactions.

The **P-Chain** processes various platform-level operations, such as creation of new blockchains and sub-networks, validator (de-)registration, or staking operations. It also uses Snowman.

The Avalanche protocol introduced in the Ava Labs whitepaper, which is also mainly marketed and presented in online materials, is used as the basis of X-Chain. Interestingly, the Snowman protocol, which supports the C-Chain and P-Chain, is almost absent from documentation and marketing, and remains outside the scope of this work.

2.1 Validators

Participants in the Avalanche protocol are called validators or nodes. Validators following the protocol are called honest. As a blockchain protocol, Avalanche aims to be resilient to validators deviating from the protocol, which are called malicious, or collectively as the adversary.

Avalanche employs a Proof-of-Stake mechanism to control the ability of malicious validators joining the system. Validators need to acquire AVAX tokens (2,000 minimum) and deposit them using the Avalanche platform to actively participate in the agreement process. Validators are associated with, and weighted by, the amounts of deposited tokens, called their *stake*. Typically, Proof-of-Stake blockchains aim to be resilient to the adversary that is able to acquire a stake smaller than $1/3$ of the total tokens (which is the theoretical maximum in harsh network conditions).

2.2 UTXO Model

Avalanche uses the Unspent Transaction Output (UTXO) model, as initially introduced in Bitcoin [12]. In the model, a transaction contains a set of inputs, a set of outputs, and a digital signature. Each input of a transaction corresponds to

a specific output from a previous transaction. Transactions are issued by users, processed by the system, and as a result are accepted or rejected by the system.

Two transactions including the same input are *conflicting*, and only one transaction from such a pair can be accepted by the system.

The balance of a user is determined by the set of outputs transferred to that user in previously accepted transactions and not yet used as inputs for newer transactions. A valid transaction is also signed with keys corresponding to the relevant inputs.

In contrast to most blockchains such as Bitcoin, Avalanche does not necessitate a total order of all transactions. Instead, transactions in Avalanche form a directed acyclic graph (DAG) resulting in a partial order. A transaction tx' depends on tx if tx' consumes an output of tx. In this case, every validator needs to process tx before processing tx'. Validators can execute transactions that are not dependent on each other in any order.

3 Snowball

The Snowball protocol serves as the foundational component of the Avalanche blockchain. It is based on continuously querying random sets of k validators regarding their current "approval" regarding a transaction, denoted as T.

When performing a query on $k = 20$ nodes within a Snowball instance, the selection probability of a node is proportional to the stake of the node. Intuitively, the influence of validators in validating transactions, quantified by the probability of them being queried, is determined by their stake.

Validators maintain a confidence value for each binary choice: Blue if they prefer to accept transaction T, Red if they reject transaction T. When a validator queried k other nodes and saw at least α for either Red or Blue, we say that this color received a *chit*, and the confidence value for that color is incremented by one. When queried, a validator will either respond Blue if the confidence value for Blue is higher, or Red if the confidence value for Red is higher. A color is accepted by a node if for at least β consecutive rounds of querying it received a chit. The logic of Snowball is illustrated in Figure 1.

3.1 Safety

Intuitively, safety properties can be understood as "bad" things not happening. In our context, the main safety property is ensuring that two honest nodes cannot perceive two conflicting transactions as accepted. The Avalanche whitepaper outlines the definition of safety as follows:

P1. Safety: When decisions are made by any two honest nodes, they decide on conflicting transactions with negligible probability ($\leq \varepsilon$).

Here ε represents the safety failure probability, with the specific value dependent on the maximum number f of adversarial nodes, which is not explicitly stated in the formal definition provided by the Avalanche whitepaper.

```
k, α, β ← 20, 15, 20                                   ▷ Protocol parameters

function SNOWBALL(V, v_self, c_init)
    c_pref ← c_init
    c_last ← c_init
    confidence ← [0, 0]
    counter ← 0

    while counter < β do
        V_query ← SAMPLEVAL(V \ {v_self}, k)    ▷ Weighted by stake with replacement.
        R ← QUERY(V_query)  ▷ Query each with v_self, c_pref; R is multiset of responses.
        for i ∈ {0, 1} do
            if |[r ∈ R | r = i]| ≥ α then
                if c_last ≠ i then
                    counter ← 0
                confidence[i] ← confidence[i] + 1
                if confidence[i] > confidence[1 − i] then
                    c_pref ← i
                c_last ← i
                counter ← counter + 1
    return c_last

function RESPONDTOQUERY(querier, c_querier)
    if c_pref = ⊥ then
        c_pref ← c_querier
    return c_pref
```

Fig. 1. Snowball algorithm.

3.2 Liveness

Liveness refers to the continued operation of the system. In our context, liveness mainly refers to ensuring that all honest nodes eventually decide to accept or reject a transaction within a reasonable time frame.

According to the whitepaper, Avalanche has the following liveness guarantees:

P2. Liveness (Upper Bound): Snow protocols terminate with a strictly positive probability within t_{max} rounds.

P3. Liveness (Strong Form): If $f \in \mathcal{O}(\sqrt{n})$, then the snow protocol terminates with high probability ($\geq 1 - \varepsilon$) in $\mathcal{O}(\log(n))$ rounds.

However, it is specified later in the whitepaper that P2 holds only under the assumption that initially, one proposal has at least $\frac{\alpha}{k}$ support in the network, for which there is no guarantee.

4 Simulation

To test resilience of Snowball, we perform a local simulation of the protocol using a Rust implementation [3]. As the base implementation of Snowball (c.f. Figure 1) we use the `avalanche-consensus` Rust library[4], which is a translation of the Snowball Go code that is part of the official Avalanche implementation[5] and is maintained by the Ava Labs team.

The simulation involves a network of multiple honest nodes executing the protocol correctly, aiming to achieve agreement on a binary decision. Malicious nodes collude to perform the considered attacks. In our experimental scenarios, the stake is equally divided among validators.

4.1 Network Assumptions

Distributed protocols might require various network reliability assumptions to work correctly. Many blockchain protocols guarantee safety in harsh network conditions, such as those of partially synchronous models, where messages can be greatly delayed.

In our simulation we make the strongest network reliability assumptions possible, where every message arrives with the same, known latency. We deny the adversary any communication advantage whatsoever, including advantages often practically achievable by an attacker in the real world, such as performing queries faster, or performing more queries.

In synchronous rounds, nodes query other nodes, as described by the Snowball protocol. Between rounds, the nodes update their preferred color with which they respond to the queries.

4.2 Adversary Information

To perform the attacks, the adversary needs information about the other nodes' preferred color. We call the adversary *naive* if the adversary simply queries honest nodes in line with the protocol and updates his estimation of the colors preferred by the honest nodes according to the query results.

We also consider an adversary that possesses accurate information about the numbers of nodes preferring Red/Blue in the current round, and call that adversary *informed*.

4.3 Liveness Attack

When attacking the liveness property, the adversary aims to delay the decision of honest nodes by keeping the network split equally between Red and Blue. The attack strategy we consider is straightforward. When the adversary is queried, it responds with the color that is less preferred among all honest validators. By

n, f ▷ *Network parameters.*
μ_{estimate} ▷ *Current estimate of network-wide preference towards 1.*

function RespondToQuery($v_{\text{query}}, c_{\text{querier}}$)
  if $\mu_{\text{estimate}} < 0.5$ **then**
    return 1
  return 0

Fig. 2. Adversary strategy for liveness attack.

doing so, the adversary aims to bring the network split between the Red and Blue decisions closer to the even 50-50 split. This is shown in Figure 2.

For a given experimental scenario, we consider the attack successful if in more than 5 out of 10 simulation runs, no validator has terminated with a decision after 100,000 rounds. We note that if a round of querying took about 1 second, 100,000 rounds would correspond to over a day.

We perform binary search with respect to the adversary stake to find the minimal fraction of total stake for which the adversary is successful. Figure 3 shows the minimum percentage of stake the adversary needs to attack the liveness of the protocol. It can be seen to decrease significantly with increasing number of total nodes in the network, showing the sub-linear security bound.

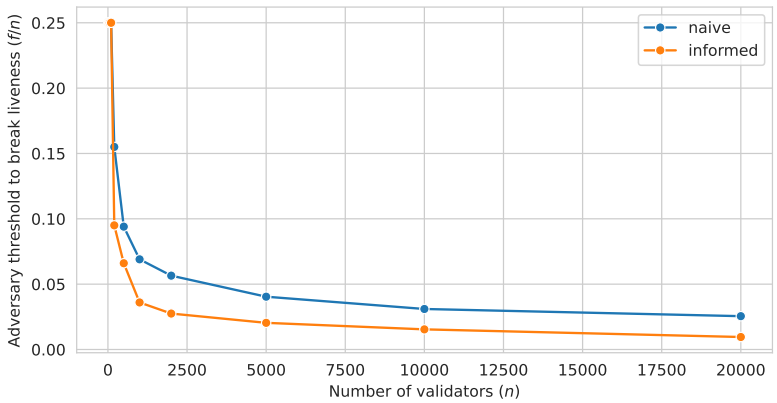


Fig. 3. Minimum fraction of stake needed by the adversary to successfully attack liveness, plotted against the number of nodes in the network with equal stake.

[4] https://crates.io/crates/avalanche-consensus
[5] https://github.com/ava-labs/avalanchego

```
n, f                                  ▷ Network parameters.
V_target                              ▷ Validators targeted in the safety attack.
μ_estimate                            ▷ Current estimate of network-wide preference towards 1.
μ_target                              ▷ Target split to maintain before finalization.
fin                                   ▷ Indicator if some targeted validator has finalized.

function RespondToQuery(v_query, c_querier)
    if fin then
        return 0
    else if v_query ∈ V_target then
        return 1

    if μ_estimate < μ_target then     ▷ Otherwise, continue with regular liveness attack.
        return 1
    return 0
```

Fig. 4. Adversary strategy for safety attack.

4.4 Safety Attack

In this scenario, the adversary aims to break the safety property of the protocol by causing some honest nodes to accept conflicting transactions.

The adversarial strategy we consider starts with maintaining a modified liveness attack. While the honest nodes are divided between Red and Blue, denote by μ the fraction of honest nodes that prefer Red. Then, the fraction of honest nodes that prefer Blue is $1 - \mu$. The adversary attempts to keep the numbers of honest nodes preferring Red and Blue close to the $\mu : \mathit{Red}$, $1 - \mu : \mathit{Blue}$ split by replying to queries with colors that sway the honest nodes towards this split. Additionally, the adversary chooses a set of honest nodes to queries of which the adversary responds exclusively with Red. By employing this approach, the attacker can significantly increase the likelihood of the targeted nodes finalizing with the color Red after some time, while at the same time keeping the rest of the network from deciding in either direction. Once some targeted node accepts Red, the adversary replies to all queries with the color Blue, such that the rest of the network accepts Blue. As a result, the adversary produces a safety violation, as the targeted node decides differently to the rest of the network.

Figure 4 describes the adversarial strategy for the considered safety attack.

4.5 Safety Attack Analysis

Consider the attack where a single node is targeted. Denote the number of adversarial nodes by f and the number of nodes in total by n. Assuming that the adversary can maintain the honest nodes split of $\mu : \mathit{Red}$, $1 - \mu : \mathit{Blue}$, in expectation we observe the following: when the targeted node queries, it receives a percentage of $\mu(1 - \frac{f}{n}) + \frac{f}{n}$ responses for Red, while other nodes receive $\geq \mu(1 - \frac{f}{n})$ fraction of responses for Red, depending on the adversary. For example, with 30% adversary stake and a split of 69.4% Red and 30.6% Blue among

honest nodes, the targeted node has a probability $p = 0.694 \cdot 0.7 + 0.3 = 0.7858$ of receiving a Red response when querying. Consequently, the targeted node can finalize Red with some probability, which eventually occurs.

Once this happens, the adversary replies to all queries with Blue. Since $\mu(1 - \frac{f}{n}) = 0.694 \cdot 0.7 = 0.4858 < 0.5$, it is very likely that all honest nodes flip to Blue and later accept Blue. In general, if $\mu(1 - \frac{f}{n}) < 0.5$, the adversary still has the ability to sway the network towards accepting Blue.

We now compute the probability that the targeted node converges to Red, given that it sees an average proportion of $\mu(1 - \frac{f}{n}) + \frac{f}{n}$ of responses in favor of Red when querying. Recall that when querying $k = 20$ other nodes, a validator increments its successive success counter, denoted as "counter", only if a color receives at least $\alpha = 15$ votes, and if this color is the same as the currently preferred color. Otherwise, the success counter is reset to 0.

Let the random variable X denote the number of participants who prefer Red in a sample of size $k = 20$. We want to calculate the probability distribution $P(X \geq \alpha) = 1 - P(X < 15)$. We can model this using a binomial distribution with parameters $p = \mu(1 - \frac{f}{n}) + \frac{f}{n}$ for the targeted node and $p = \mu(1 - \frac{f}{n})$ for the other nodes. Thus, we have:

$$P(X \geq \alpha) = 1 - P(X < \alpha) = 1 - F(\alpha - 1, k, p) = 1 - \sum_{i=0}^{\alpha-1} \binom{k}{i} p^i (1-p)^{k-i}$$

Here, $F(\alpha - 1, k, p)$ represents the cumulative distribution function (CDF) of the binomial distribution. For our previous example, where $\mu = 0.694$, for honest nodes other than the attack target, we can calculate the probability $P(X \geq \alpha)$, which represents the chance of reaching the α majority threshold for the color Red when querying. Plugging in the probability to receive a response supporting Red in a single round $p = 0.4858$ from above, we get $P(X \geq \alpha) = 1 - F(14, 20, 0.4858) \approx 0.015$. On the other hand, the targeted node that has a probability $p = 0.7858$ to get a Red response, and so $p_\alpha = P(X \geq \alpha) = 1 - F(14, 20, 0.7858) \approx 0.756$. This means that our targeted node has a $p_\alpha \approx 75.6\%$ chance of reaching the α majority for Red when querying $k = 20$ other nodes, whereas other honest nodes only have a 1.5% chance of the same. Consider the expected number of iterations needed to obtain $\beta = 20$ consecutive successes of reaching the $\alpha = 15$ majority for a color. Let X_β represent the number of trials required to achieve β consecutive successes, with the probability of one success being p_α. From [8], we can use the following formulas:

$$\mathbb{E}[X_\beta] = \frac{1 - p_\alpha^\beta}{(1 - p_\alpha) p_\alpha^\beta}$$

$$\mathbb{V}ar[X_\beta] = \frac{1 - (2\beta + 1)(1 - p_\alpha) p_\alpha^\beta - p_\alpha^{2\beta+1}}{(1 - p_\alpha)^2 p_\alpha^{2\beta}}$$

For the example where $p_\alpha = 0.756$ for the targeted node, we obtain:

$$\mathbb{E}[X_{20}] \approx 1095$$

$$\sigma = \quad \overline{\mathbb{V}ar[X_{20}]} \approx 1078$$

On average, the targeted node needs to query 1,095 times with a standard deviation of 1,078. We confirmed the expected results experimentally.

Adversary	Honest Split	p_α	$\mathbb{E}[X_{20}]$	σ	$\approx \mathbb{E}[X_{20}^{1000}]$
30%	69.4% - 30.6%	0.756	1,095	1,078	20
25%	64.8% - 35.2%	0.560	245,562	245,544	265
20%	60.7% - 39.3%	0.364	9.4e+8	9.4e+8	940,000
10%	54.0% - 46.0%	0.101	8.4e+19	8.4e+19	8.4e+16
5%	51.2% - 48.8%	0.043	2.5e+27	2.5e+27	2.5e+24

Table 1. Summary of the expected safety attack results for different percentages of adversarial stake and corresponding stable network splits. The second column shows the maximally imbalanced but stable split of honest validators that the adversary is able to maintain. $\mathbb{E}[X_{20}^{1000}]$ is the expected length of the safety attack when 1000 nodes are targeted.

The effectiveness of the attack can be greatly increased by targeting a large number of nodes rather than just one. Let X_β^k be the number of trials required for any target node among k targeted nodes to achieve β consecutive successes. Assuming the β successes to be equally probable to conclude in every round after 19, and assuming the constant network split maintained by the adversary, the expected number $\mathbb{E}[X_\beta^k]$ is $\frac{\mathbb{E}[X_\beta - 19]}{k} + 19$. We have simulated some experimental scenarios, such as targeting 1000 nodes with $n = 3000$ and the adversarial stake of $f = 750$, where the results matched our expectation.

With increasing total number of nodes n, the adversary can target more nodes. While in our experiments we successfully targeted over $0.3n$ of $n = 3000$ nodes, future work is needed to understand how big the share of targeted nodes can be in an optimal strategy and with increasing n. In summary, the strength of the attack corresponds exponentially to the share of the adversary stake. On the other hand, the expected required duration of the attack is inversely linear to the overall number of nodes n, as the number of targeted nodes can increase roughly linearly with n. Table 1 summarizes the effectiveness of the safety attack for adversaries of different strengths.

5 Avalanche Protocol

In this section, we explain how the Avalanche protocol builds on Snowball to incorporate optimizations and additional features.

5.1 DAG

To enhance the throughput and enable parallel processing of transactions, the Avalanche protocol builds a directed acyclic graph (DAG) for transactions, in-

stead of a linear chain. Each transaction is represented as a node in the DAG. Furthermore, transactions in the DAG are interconnected through parent-child relationships: A transaction T refers to older transactions known as its parents $\mathsf{Parents}(T)$. We denote the parent relation $T' \in \mathsf{Parents}(T)$ by $T' \leftarrow T$. If T'' is reachable by parent links from T, we say that T'' is an ancestor of T, or $T'' \in \mathsf{Ancestors}(T)$, and that T is a descendant of T'', or $T \in \mathsf{Descendants}(T'')$.

5.2 Vertex

In order to limit the coordination overhead, a node in the Avalanche DAG is not an individual transaction but rather a batch of transactions known as a *vertex*. A vote for a vertex is considered a vote for all transactions contained within that vertex. This allows Avalanche to facilitate efficient queries, while still maintaining confidence levels and a conflict set for each individual transaction.

When a vertex is accepted, all transactions within it are accepted. When a vertex is rejected, valid transactions in that vertex may be batched into a new vertex, by removing the non-preferred transactions that resulted in the vertex getting rejected. When a node creates a vertex V, it chooses parents for V that are currently preferred.

When a user submits a payload transaction tx, a node creates a transaction $T\langle tx, \mathcal{D}\rangle$ for that payload. It includes the payload tx, along with the set of UTXO IDs that will be consumed if the transaction is accepted, and the list $\mathcal{D}$ of dependencies on which this transaction relies. Each dependency must be accepted before this transaction can be accepted. The node then batches this transaction $T\langle tx, \mathcal{D}\rangle$ with other pending transactions into a vertex. The node assigns one or more parents to this vertex, allowing it to be added to the DAG. We define an Avalanche transaction T as preferred if it is the preferred transaction in its conflict set P_T. In other words, if transaction T has the highest confidence among other conflicting transactions. Each node u calculates the confidence value for each transaction T denoted by $d_u[T]$. This confidence value is defined as the sum of the chits received by T and all its descendants [13]: $d[T] = \sum_{T' \in \mathcal{T}_u : T \in \mathsf{Descendants}(T')} c_{u,T'}$. Here, $\mathcal{T}_u$ represents all the transactions currently known by node u in its view of the DAG, and $c_{u,T'}$ represents the chit received by transaction T'. $c_{u,T}$ can only take two values: 0 or 1. Node u queries transaction T only once, as the votes on the descendants of T also serve as queries and votes on T. Specifically:

$$c_{u,T} = \begin{cases} 1 & \text{transaction } T \text{ received a chit when } u \text{ queried for it} \\ 0 & \text{otherwise} \end{cases}$$

As a reminder, receiving a chit for transaction T means that node u received an approval rate of at least $\alpha = 15$ when it queried $k = 20$ other nodes to determine if T was their preferred transaction. The confidence value of T (and thus its status as accepted or rejected) is then updated based on the queries made on its descendants.

We say that a transaction T is strongly preferred if T is preferred and all its ancestors are also preferred in their respective conflict sets. An Avalanche transaction T is considered virtuous if it conflicts with no other transactions or if it is strongly preferred. Consequently, a virtuous vertex is a vertex where all its transactions are virtuous. Similarly, a preferred or strongly preferred vertex is one where all its transactions are preferred or strongly preferred, respectively. The parents of a vertex are randomly chosen from the *virtuous frontier* set $\mathcal{VF}$, which consists of the vertices at the frontier of the DAG that are considered virtuous:

$$\mathcal{VF} = \{\ T \in \mathcal{T}\ \mid \mathsf{virtuous}(T)\ \wedge \neg\ \mathsf{virtuous}(T')\ \forall T' \in \mathcal{T} : T \leftarrow T'\}$$

The notation $\mathsf{virtuous}(T)$ indicates that T is virtuous. In other words, $\mathcal{VF}$ is the set of vertices that are virtuous, and have no virtuous children.

5.3 From Snowball to Avalanche

The Avalanche protocol runs a Snowball instance on the conflict set of each transaction T once a node hears about a new transaction that gets appended to the DAG. This means that when a new transaction T is received, a validator will query k other random nodes to determine if T is their preferred transaction. The queried nodes will respond positively only if transaction T and its ancestors in the DAG are also their preferred transactions within their respective conflict sets. Instead of querying a Snowball instance for each individual transaction, Avalanche batches transactions into a vertex and instantiates a Snowball instance for that vertex, checking if all the transactions within that vertex and its ancestors are valid.

When a node is queried about the preference of transaction T and its ancestors, it provides not just a binary vote as in Snowball, but rather responds with its entire virtuous frontier $\mathcal{VF}$ based on its local view. This allows the respondents to specify which ancestors are not preferred if T is not strongly preferred. The querying node u collects the virtuous frontier of the k queried nodes. For each virtuous frontier $\mathcal{VF}'$ sent by a node w as a vote, we add the transactions T' from $\mathcal{VF}'$ and the ancestors of T' to a set $\mathcal{G}[T, w]$, which represents the positively reported transactions of w when asked to vote for T. We then count how many times node w, when queried for T, has acknowledged a transaction T' as virtuous, and store this in the counter $ack[T, T']$. We then run a Snowball instance for every $ack[T, T']$: If $ack[T, T']$ received more than α votes it indicates that the α majority of the k queried validator agree that T' is preferred. We then increase the consecutive counter for T' if it was also the preferred transaction in the last vote. The above procedure of voting on a vertex containing a single transaction can be generalized for vertices containing multiple transactions.

Finally, there are two ways in which a vertex V, and consequently all the transactions it contains $T \in V$, can be accepted, provided that all the ancestors of V have also been accepted. The first way is if none of its transactions $T \in V$ conflict with any other transactions, and the vertex V received β_1 consecutive

successes. In this case, the vertex and all its transactions are accepted by node u. The second way is if some transactions $T \in V$ have other transactions in their conflict sets, and the vertex V receives β_2 consecutive successes. In this case, the node accepts the vertex V and all its transactions. The Avalanche protocol denotes β_1 as `betaVirtuous` and β_2 as `betaRogue`, and naturally $\beta_1 < \beta_2$.

5.4 Liveness Attack

Suppose that two transactions (both with accepted virtuous ancestors) T batched in vertex V and T' batched in vertex V' are conflicting. Recall that the Snowball liveness attack consisted of a strategy where the adversary tried to ensure that the split between Red and Blue was always close enough to 50% each. Here, the approach is similar, except that we have to ensure that, on average, 50% of the network has V in their virtuous frontier or as an ancestor of their virtuous frontier, and the other 50% of the nodes have V' in their virtuous frontier or as an ancestor of their virtuous frontier. The binary attack can be transposed to one where the adversaries responds with the virtuous frontier $\mathcal{VF}$, with V an ancestor of the nodes in $\mathcal{VF}$, or responds with the virtuous frontier $\mathcal{VF}'$, with V' an ancestor of the nodes in $\mathcal{VF}'$. The intuition behind this attack is that half of the nodes will adopt a virtuous frontier that contains vertex V as a virtuous vertex, and the other half of the nodes will adopt a virtuous frontier that contains V' as a virtuous node. At every iteration of the loop, the adversary needs to maintain those two conflicting virtuous frontiers $\mathcal{VF}$ and $\mathcal{VF}'$, grow the DAG such that some new valid vertices are appended to the conflicting $\mathcal{VF}$ and $\mathcal{VF}'$, and respond accordingly with either one of the virtuous forests using the same technique that was used for the Snowball liveness attack.

5.5 Safety Attack

The safety attack from Snowball to Avalanche can be transposed in the same way as was explained above for the liveness attack. Similarly, we will try to maintain a network split that does not converge: μ of nodes will prefer a virtuous frontier $\mathcal{VF}$ that contains v as an ancestor, and ν of nodes will prefer a virtuous frontier $\mathcal{VF}'$ that contains v' as an ancestor. For one targeted node, the adversary will respond exclusively with the virtuous frontier $\mathcal{VF}$ instead of trying to maintain a split. This way, we can make the targeted node to accept vertex v while all the other nodes are still undecided. Once this is done, the adversary can unanimously respond with virtuous frontier $\mathcal{VF}'$ to make the rest of the nodes accept v' in order to break safety. Such an attack can be instantiated by any adversary that creates conflicting (double spending) transactions T and T', batch them in nodes v and v' and conducts the attack to make some nodes accept T, and some other nodes accept T' thus resulting in a successful double spending.

6 Consensus or Broadcast

Consensus is a property that allows multiple parties to reach agreement on transactions, either accepting or rejecting them. In the context of blockchain systems, consensus can be defined by the following set of properties:

Definition 1. *Each honest validator observes some transaction from a set of conflicting transactions* $\{t_0, t_1, \dots\}$. ***Consensus*** *satisfies the following properties:*

Totality*: If some honest validator accepts a transaction, every honest validator will eventually accept the same transaction.*
Agreement*: No two honest validator accept conflicting transactions.*
Validity*: If every honest validator observes the same transaction (there are no conflicting transactions), this transaction will be accepted by all honest validators.*
Termination: *Some transaction from the set will eventually be accepted by honest validators.*

As implied by Agreement and Termination, a consensus protocol enables nodes to reach an agreement on conflicting transactions, where multiple valid transactions consuming the same input are involved. In such cases, all nodes should unanimously accept one of the conflicting transactions.

As we have established, the Avalanche protocol features a relatively weak, sublinear resilience to liveness attacks involving conflicting transactions. To address this issue, Avalanche introduces the term of virtuous transactions, which can enjoy better guarantees. In other words, even for a relatively small adversary, Avalanche does not satisfy the Termination property, and only guarantees termination if the Validity condition is also met: all honest validators observe just one valid transaction and no conflicting ones.

The termination property becomes crucial in scenarios involving smart contracts, where conflicting transactions may arise, such as two users attempting to purchase the same product. To address this limitation, the Avalanche team introduced a different solution for the C-Chain and P-Chain, specifically designed to execute smart contracts required for such blockchain applications.

As described by [9], consensus is not necessary for payment systems, and indeed there exist payment systems providing similar guarantees to Avalanche, while also unable to support general applications such as smart contracts: broadcast-based payment systems [4,7,5,15,11]. The provided guarantees of a Byzantine reliable broadcast can be defined as follows:

Definition 2. *Each honest validator observes some transaction from a set of conflicting transactions* $\{t_0, t_1, \dots\}$. ***Byzantine reliable broadcast*** *satisfies the properties of Consensus, without the Termination property.*

Thus, referring to Avalanche as a consensus protocol can be misleading, as it is more akin to broadcast-based payment systems. While the performance of Avalanche is given prominence, a different solution has been used as required by the C-Chain and P-Chain.

7 Related Work

Amores-Sesar et al. [2] analyze the Avalanche protocol. They explain the protocol with pseudocode and introduce a property of Avalanche that was omitted here: No-op transactions which are stateless transactions are added into the DAG by the nodes to make sure we always make progress on the finalization of older transactions. The paper introduces a liveness attack (different from ours) that could be possible if the naive way of voting for a transaction and its ancestors with just a binary yes/no vote was implemented. However, this is not the case, as emphasized at the end of the paper with the pseudocode involving the virtuous frontier concept.

Ash Ketchum and Misty Williams [10] raise concerns similar to ours in their recent write-up, that Avalanche is not a consensus protocol.

Most recently, a follow-up analysis by Amores-Sesar et al. [1] formalized the need for at least $\Omega(\log n + \beta)$ rounds for consensus with the Snow family of protocols. They then proposed a specific modification of Snowflake and Snowball implementing this change.

8 Conclusion

In this paper, we have examined the resilience properties of Avalanche and its underlying Snowball protocol. We have experimentally evaluated simple strategies for a potential adversary. To quantify the efficacy of these attacks, we have conducted simulations and evaluated the ratio of stake the adversary needs to control to launch successful attacks on liveness and safety.

Our analysis revealed that an adversary with a small fraction of the stake can indefinitely keep the network in a state where it cannot finalize a transaction. With some probability depending on the stake and the size of the network, the adversary can also convince some node to finalize a transaction that is then rejected by other honest parties, which can result in a double spending attack.

Through our analysis, we have demonstrated that the Snowball protocol - the foundation of Avalanche - is vulnerable, when conflicting transactions are present. The weak resilience when conflicting transactions are present is a critical limitation, as it makes the protocol unable to support general smart contracts. This explains why Avalanche actually uses a different protocol, called Snowman, which uses a linear blockchain (instead of a DAG) in order to totally order those transactions, unlike what is done for payments [14].

Future Work The basis of our attacks relies on the presence of conflicting transactions. Future work could analyze how Avalanche distinguishes unique transactions, and determine the feasibility for an adversary to arbitrarily create conflicting transaction from another transaction T broadcast by an honest node, for example, by creating a copy with different parents in the DAG.

References

1. Amores-Sesar, I., Cachin, C., Schneider, P.: An analysis of Avalanche consensus. In: International Colloquium on Structural Information and Communication Complexity. pp. 27–44. Springer (2024)
2. Amores-Sesar, I., Cachin, C., Tedeschi, E.: When is spring coming? a security analysis of Avalanche consensus. arXiv preprint arXiv:2210.03423 (2022)
3. Anonymized for double-blind review: Git repository (2024)
4. Baudet, M., Danezis, G., Sonnino, A.: Fastpay: High-performance byzantine fault tolerant settlement. In: Proceedings of the 2nd ACM Conference on Advances in Financial Technologies. pp. 163–177 (2020)
5. Baudet, M., Sonnino, A., Kelkar, M., Danezis, G.: Zef: low-latency, scalable, private payments. In: Proceedings of the 22nd Workshop on Privacy in the Electronic Society. pp. 1–16 (2023)
6. Buterin, V.: Ethereum: A next-generation smart contract and decentralized application platform (2014), `https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf`
7. Collins, D., Guerraoui, R., Komatovic, J., Kuznetsov, P., Monti, M., Pavlovic, M., Pignolet, Y.A., Seredinschi, D.A., Tonkikh, A., Xygkis, A.: Online payments by merely broadcasting messages. In: 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). pp. 26–38. IEEE (2020)
8. Drekic, S., Spivey, M.Z.: On the number of trials needed to obtain k consecutive successes. Statistics & Probability Letters **176**, 109132 (2021)
9. Guerraoui, R., Kuznetsov, P., Monti, M., Pavlovič, M., Seredinschi, D.A.: The consensus number of a cryptocurrency. Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing - PODC '19 (2019)
10. Ketchum, A., Williams, M.: On pseudo-profound bullshit in the avalanche whitepaper (2019)
11. Mathys, M., Schmid, R., Sliwinski, J., Wattenhofer, R.: A Limitlessly Scalable Transaction System. In: 6th International Workshop on Cryptocurrencies and Blockchain Technology (CBT), Copenhagen, Denmark (September 2022)
12. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008), `https://bitcoin.org/bitcoin.pdf`
13. Rocket, T., Yin, M., Sekniqi, K., van Renesse, R., Sirer, E.G.: Scalable and probabilistic leaderless BFT consensus through metastability. arXiv preprint arXiv:1906.08936 (2019)
14. Sekniqi, K., Laine, D., Buttolph, S., Sirer, E.G.: Avalanche platform. Netw. Distrib. Ledgers (2020)
15. Sliwinski, J., Wattenhofer, R.: Asynchronous proof-of-stake. In: 23rd International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS) (November 2021)

We will DAG you

Ignacio Amores-Sesar[1][0000−0002−1751−1515] and
Christian Cachin[1][0000−0001−8967−9213]

University of Bern, Neubrückstrasse 10, 3012 Berne, Switzerland
{ignacio.amores, christian.cachin}@unibe.ch

Abstract. Protocols based on directed acyclic graphs (DAG) have been proposed as potential solution to the latency and throughput limitations of traditional consensus protocols. However, their adoption has been hindered by security concerns and a lack of a solid foundation to guarantee improvements in both throughput and latency. In this paper, we present a construction that rigorously demonstrates how DAG-based protocols can achieve superior throughput and latency compared to chain-based consensus protocols, all while maintaining the same level of security guarantees.

Keywords: Consensus · Chain · DAG · Security · Latency · Throughput.

1 Introduction

In the ever-evolving landscape of distributed systems, achieving consensus among a set of processes has become a fundamental challenge that has garnered significant attention in recent years. Consensus protocols are a universal primitive in distributed computing, ensuring that a network of interconnected processes can collectively agree on a shared state despite potential failures or malicious actors. However, as the demands on distributed systems continue to grow, the need for consensus protocols that can deliver both higher throughput and lower latency has become increasingly pressing. This need is particularly relevant in permissionless consensus protocols as used by cryptocurrencies and blockchain protocols, which face stringent demands on their throughput and latency.

Traditional consensus protocols have exhibited considerable advancements in both throughput and latency since the first practical consensus protocols [12,7]. One of the most promising lines of work are DAG consensus protocols as introduced by the "All you need is DAG" paper [10] and subsequently extended by Narwhal and Tusk [8], Bullshark [22], and Cordial Miners [11]. A common characteristic of these protocols is their capacity to enable every participant to generate blocks that reference previous blocks, forming a *directed acyclic graph* (*DAG*). In permissionless protocols like Bitcoin [14], every process (miner) can create a block upon successfully solving the cryptographic puzzle. Therefore, the concept of constructing a DAG that is later ordered, as proposed by Keidar et

al. [10], holds the potential to enhance the throughput and latency of permissionless consensus protocols. In essence, DAG protocols may surpass traditional permissionless consensus protocols, which form a chain.

The evident approach to improving the throughput of chain protocols is to increase the block ratio, i.e., the number of blocks produced per unit of time, effectively accelerating the execution of the protocol as there is less time between created blocks. This goal can be pursued by lowering the difficulty in *Proof-of-Work* (*PoW*) protocols. However, increasing the block ratio may harm the protocol since it elevates the likelihood of forks—situations where two different processes create blocks extending the chain. An abandoned block is one that is never output by the protocol; whenever a chain protocol forks, an abandoned block is produced. Therefore, despite the increased number of generated blocks, the number of abandoned blocks concurrently rises, adversely affecting the protocol's throughput. Moreover, it is imperative to recognize that the block ratio cannot be augmented arbitrarily without compromising the protocol's security.

In this paper, we introduce a construction that takes as input a DAG-based protocol or a chain protocol Π, which may produce abandoned blocks and produces a new DAG protocol Π' with the property that every created block is eventually output. Specifically, Π' creates the same number of blocks as the base protocol Π and outputs *every* created block of Π. We show that the safety and liveness of Π' reduces to the safety and liveness of Π. In simpler terms, Π' is as safe and live as Π. Furthermore, we establish that Π' has lower or equal *latency* as Π while achieving strictly higher *throughput*. Our main contribution lies in a formal proof that chain protocols cannot achieve optimal throughput, i.e., for any chain protocol Π, there is a DAG protocol Π' that is safe and life under the same assumptions as Π, with the same or better latency and better throughput.

2 Related work

DAG protocols represent a recent breakthrough within the domain of permissioned consensus protocols [10,8,22,11]. While DAG protocols have been previously introduced in the permissionless context, their adoption and success have been somewhat restrained due to their inherent complexity when compared to traditional chain protocols. Several well-known DAG protocols have exhibited vulnerabilities, highlighting challenges in their success. For instance, IOTA [18], one of the pioneering DAG protocols, has been susceptible to vulnerabilities such as Parasite-chain attacks [18,17]. Another promising protocol, GhostDAG [20], has also revealed vulnerabilities in its design [13]. Even Avalanche [19], the most successful DAG protocol in terms of market capitalization, originally had vulnerabilities in its design [3].

An intriguing DAG protocol to note is Conflux[13], which leverages the GHOST consensus rule [21] and augments blocks with additional references to transform a chain protocol into a DAG. Li et al. [13] have demonstrated that Conflux's security is directly inherited from the security of GHOST. However,

it is worth mentioning that the GHOST protocol has exhibited lower resilience than other consensus protocols in the presence of network malfunctions [15,4].

Our contribution to this landscape is a formal proof of the superior performance of DAG protocols, facilitated by a construction that can be conceptualized as an extension of the Conflux construction [13]. Specifically, when we instantiate the throughput closure using GHOST [21], we arrive at Conflux [13].

3 Abstractions

We consider a set of n *processes* $\mathcal{P} = \{P_1, P_2, \ldots\}$ that interact with each other by exchanging messages through the network. A protocol Π for $\mathcal{P}$ consists of a collection of programs with instructions for all processes. In particular we are interested in the study of *chain protocol* and *DAG protocol* protocols, i.e., protocol that rely on a chain or a DAG to deliver blocks. These two concepts are formally defined below.

Chain and DAG protocols are pivotal tools employed to establish robust and secure ledgers, and as such, they must adhere to specific fundamental requirements.

Traditionally, the gold standard concept is *atomic broadcast* [6], which ensures that all processes deliver the same set of transactions in the same order. In this paper, we consider a variant of this abstraction that includes the concept of a *block* in the interface and properties [2]. Processes broadcast transactions and deliver blocks using the events *bab-broadcast*(tx) and *bab-deliver*(b), respectively, where block b contains a sequence of transactions $[tx_1, \ldots, tx_m]$. The protocol outputs an additional event *bab-mined*(b, P), which signals that block b has been *mined* by process P, where P is defined as the *miner* of b. The event *bab-mined*(b, P) can be understood as the creation of block b by process P. Notice that *bab-mined*(b, P) signals only the creation of a block and not its delivery. In addition to predicate $VT()$ that determines the validity of a transaction, we also equip our protocol with a validity predicate $VB()$ to be applied to blocks. These predicates and function are determined by the higher-level application or protocol.

Definition 1. *A protocol implements* block-based atomic broadcast *with validity predicates $VT()$ and $VB()$ if it satisfies the following properties, except with negligible probability:*

Validity: *If a correct process invokes a* bab-broadcast(tx)*, then every correct process eventually outputs* bab-deliver(b)*, for some block b that contains tx.*

No duplication: *No correct process outputs* bab-deliver(b) *more than once.*

Integrity: *If a correct process outputs* bab-deliver(b)*, then it has previously output* bab-mined$(b, \cdot)$ *exactly once.*

Agreement: *If some correct process outputs* bab-deliver(b)*, then eventually every correct process outputs* bab-deliver(b)*.*

Total order: *Let b and b' be blocks, and P_i and P_j correct processes that both output* bab-deliver(b) *and* bab-deliver(b')*. P_i delivers b before b' if and only if P_j delivers b before b'.*

External validity: *If a correct process outputs bab-deliver*(b)*, then* $VB(b) =$ TRUE.

The block-based atomic broadcast abstraction can be implemented by protocols based on different approaches. These differences are not captured in Definition 1, but can still be relevant for the performance of the protocol. The two families of protocols of interest for this paper are *chain protocol* and *DAG protocol* protocols. The distinguishing factor between them lies in the set of references to previously mined blocks. Specifically, for a given block b, we denote the set of *bab-mined* blocks referenced by b as *parents*(b), commonly known as the *parents* of b. Furthermore, the set of *bab-mined* blocks reachable through references from b is represented as *ancestors*(b) and is often referred to as the *ancestors* of b. A block b is a *descendant* of its ancestors. A block with no descendants is also called *leaf*.

Definition 2 (Chain protocol, DAG protocol). *A block-based atomic broadcast protocol* Π *is a* DAG protocol *protocol if* Π*-mined blocks contain references to other* Π*-mined blocks, meaning that the set of references is not empty.* Π *is a* chain protocol *protocol if every* Π*-mined block refers to exactly one* Π*-mined block and for every honest process* P_i *there is a* Π*-delivered block* b *such that every* Π*-delivered by* P_i *is* b *or in* ancestors(b)*. In essence,* Π*-delivered blocks form a chain.*

Figure 1 illustrates an example of both chain and DAG protocols.

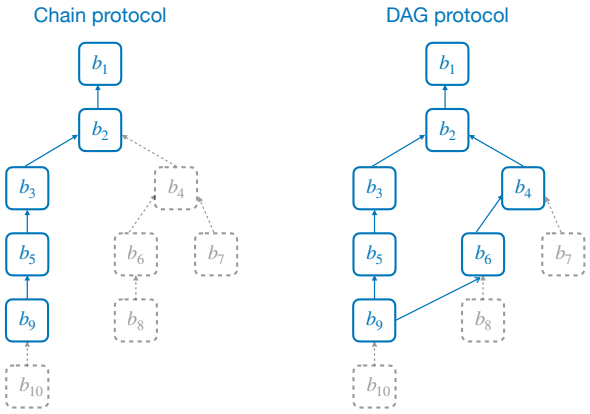


Fig. 1. Comparison between a chain protocol and a DAG protocol. Blocks in blue (continuous lines) are the *bab-delivered* blocks, whereas grey (dashed) blocks are *bab-mined* but not *bab-delivered*. The protocol on the left is a chain protocol; each block refers to exactly one block, and there is a block (b_9) such that every currently *bab-delivered* block is b_9 or an ancestor of it. The protocol on the right is a DAG protocol; block b_9 references multiple blocks.

To set the stage, we make the assumption that both chain and DAG protocols begin with an initial, hard-coded block referred to as the *genesis* block. This genesis block is special in that it possesses an empty set of references. It is important to note that, according to Definition 2, chain protocols inherently are DAG protocols. The blocks mined in chain protocols produce a *tree*, a particular kind of DAG. Therefore, for the remainder of this paper, we will use the term "DAG protocol" to encompass both DAG protocol and chain protocols, acknowledging this inclusion.

One significant implication of abstracting DAG protocols as block-based atomic broadcast (Definition 1) is that the protocol must define a function that operates on DAG that produces a list of delivered blocks. It is worth mentioning that certain DAG protocols, such as the original Avalanche protocol [19,3], do not output an ordered list of transactions but the list output by different processes may differ up to permutation. While DAG protocols can also be modeled as generic broadcast [16], situations arise where complete transaction ordering, as seen in calls to smart contracts, becomes necessary. For the purposes of this paper, we focus on protocols that can be effectively modeled as block-based atomic broadcast, i.e, protocols that achieve total order. The results we derive in this context generalize straightforwardly to protocols modeled as generic broadcasts.

4 Model

DAG protocols base their security on different techniques such as *proof of work* (PoW), *proof of stake* (PoS) [9], *proof of space-time* ($PoST$) [1], or *proof of elapsed time* ($PoET$) [5]. For the sake of simplicity, we use the PoW terminology. Nevertheless, our model does not use explicit properties of PoW and, thus, includes all these techniques.

Processes. Consistent with prior research, our protocol operates without explicit knowledge of the number or identities of the processes. The processes themselves remain unaware of these details as well. We assume a static network consisting of n processes, where up to f processes are to be corrupted by the adversary, thereby exhibiting arbitrary behavior. We do not assume any general bound on f. The construction introduced in this paper takes a protocol Π as input; we leave the bound on f vary based on the input protocol Π.

Blocks. A transaction tx, comprises a set of *inputs*, a set of *outputs*, and a collection of digital signatures, as in Bitcoin [14]. Transactions have size $|tx|$, and they are grouped into blocks, as introduced in Definition 1. Each block encompasses a specific number of transactions, denoted as m, a number of references to previously *bab-mined* blocks, quantified as n_{refs}, and further parameters essential for the proper execution of protocol Π. It is noteworthy that the size of a reference, represented as $|ref|$, is significantly smaller than that of a transaction, for simplicity, we consider it to be negligible. We reiterate that protocol Π defines external validity predicates, $VT()$ and $VB()$, responsible for determining the *validity* of a transaction or block.

Network. A *diffusion functionality* implements communication among the processes, which is structured into *synchronous* rounds. The functionality keeps a distinct $RECEIVE_i$ string for each process P_i and makes it available to P_i at the start of every round. The purpose of the string $RECEIVE_i$ is to serve as a repository for all the messages received by P_i.

When a process, say P_i, instructs the diffusion functionality to *broadcast* a set of messages, it signifies that P_i has "completed its round". In response, the functionality marks P_i as having completed its operations for that specific round. The adversary, whose actions are described in detail below, possesses the ability to access the string of any process at any point during the execution. Additionally, the adversary can observe every message broadcast by any process instantaneously. Furthermore, the adversary has the capability to insert messages directly and selectively into $RECEIVE_i$ for any process P_i, ensuring that only P_i receives the message at the outset of the following round. This behavior models what is often termed a *rushing* adversary.

Once all non-corrupted processes have concluded their respective rounds, the diffusion functionality aggregates all messages that were broadcast by non-corrupted processes during that round. These aggregated messages are then appended to the $RECEIVE_i$ strings for all processes, this is the reason of the name *synchronous* rounds. Subsequently, each non-corrupted process updates its local view at the conclusion of every round. If a non-corrupted process *Π-mines* a block in round r, all processes receive the *Π-mined* block by the subsequent round $r + 1$.

Furthermore, even if the adversary causes a block to be received selectively by only some non-corrupted processes in round r, the block is received by all non-corrupted processes by round $r+2$. The update of the local view also encompasses the *Π-delivery* of blocks that meet given criteria defined by protocol Π.

Adversary. The adversary can corrupt up to f processes at the beginning of the execution. These corrupted processes may deviate arbitrarily from the protocol, adhering to the instructions from the adversary. Additionally, the adversary wields control over the *diffusion functionality*. The adversary can schedule the delivery of messages, read the contents of the $RECEIVE_i$ string for every process at any point during the execution, and directly write messages into the $RECEIVE_i$ of any process. The adversary signals the conclusion of her round by transmitting a specially designated message.

Round structure. At the beginning of the round, process P_i reads the messages in its input string $RECEIVE_i$. Then, P_i proceeds to update its internal state in accordance with the received messages and performs a set of actions defined by protocol Π. Such actions include the *Π-delivery* of blocks. P_i concludes the round by broadcasting a set of messages to the other processes.

4.1 Abandoned blocks

Definition 3. *An* execution *is a history with an entry for each round containing the actions, a list of received messages, and a list of sent messages by each process in that round.*

Since executions are not bounded, an event may be theoretically possible, but its occurrence might have a probability of zero. For instance, consider an algorithm that continuously flips an unbiased coin indefinitely. There could be an execution where all outcomes are heads, but the probability of this specific sequence of events happening is zero, as it is the limit of an infinite execution.

To circumvent these issues, we introduce the concept of a *partial execution*.

Definition 4. *Given a protocol Π, the set of* λ-partial executions Φ_λ *is defined to be the set of λ-prefixes of all executions of protocol Π. A* partial execution *is an execution that belongs to Φ_λ for some $\lambda \in \mathbb{N}$.*

Definition 5. *Given an execution $\mathcal{E}$ of a block-based atomic broadcast protocol Π, an* abandoned *block in $\mathcal{E}$ is is an honestly* bab-mined *block b such that b is not* bab-delivered *in $\mathcal{E}$.*

It is important to note that the validity property defined in block-based atomic broadcast (Definition 1) does not guarantee that every *bab-mined* block will eventually be *bab-delivered*. Instead, this property ensures that for each *bab-broadcast* transaction, there exists at least one *bab-delivered* block that contains it. The concept of abandoned blocks is a significant concern in the context of such protocols. Abandoned blocks have been honestly *bab-mined* but are never *bab-delivered*. The existence of abandoned blocks can severely impact the performance of a chain protocol or DAG protocol.

Definition 6. *A protocol Π* permits abandoned blocks *if there exist a block b and a partial execution $\mathcal{E}$ such that: b is abandoned in any extension of $\mathcal{E}$.*

Remark 1. Note that given a protocol that permits abandoned blocks, the probability, taken over the randomness of the protocol, of having at least one abandoned block in execution is greater than zero since partial executions happen with non-zero probability.

Determining whether a given protocol Π permits abandoned blocks or not can be a challenging task and, in some cases, may not be computable due to the need to simulate potentially infinitely long executions. However, for certain protocols like Bitcoin [14], the existence of abandoned blocks is a direct consequence of forks occurring among honest miners. This phenomenon is formalized in the following definition.

Definition 7. *Given an execution $\mathcal{E}$ of a given protocol Π, a round r* forked *if protocol Π outputs two events* bab-mined(b, P_i) *and* bab-mined(b', P_j) *in round r at two distinct honest processes P_i and P_j. A protocol with a forked round in at least one partial execution is a* forkable protocol.

Lemma 1. *A forkable chain protocol Π permits abandoned blocks.*

Proof. Given a forkable protocol Π, there exist a round r in which two different honest processes output events *bab-mined*(b, P_i) and *bab-mined*(b', P_j). In particular, $b \neq b'$ because their miners are different. Π is also a chain protocol. thus both b and b' have a unique reference to previously *bab-mined* blocks, so they cannot reference each other. Another implication of Π being a chain protocol is that at any point in the execution of the protocol, there exists a *bab-mined* block b^* such that every *bab-delivered* is in *ancestors*(b^*). Since every block only contains a single reference and b and b' do not refer to each other, we conclude that no honest processes can *bab-deliver* both b and b' simultaneously.

Transactions that were originally included in abandoned blocks must be re-included in subsequent blocks to maintain the validity property (Definition 1). This re-inclusion consumes space in new blocks and has implications for both latency and throughput, as we formalize below.

4.2 Throughput and latency

Definition 8. *Given a block-based atomic broadcast protocol Π, an adversary $\mathcal{A}$, and an execution $\mathcal{E}$, we define the* throughput *of Π in the presence of $\mathcal{A}$ in execution $\mathcal{E}$ as the average number of* bab-delivered *blocks per round, and we denote by throughput$(\Pi, \mathcal{A}, \mathcal{E})$.*

Definition 9. *Given a block-based atomic broadcast protocol Π, the* throughput of Π *is defined to be throughput$(\Pi) := \inf_{\mathcal{A}} \mathrm{E}[\textit{throughput}(\Pi, \mathcal{A}, \mathcal{E})]$, i.e., the infimum over all the possible adversaries $\mathcal{A}$ of the average over the randomness Π of throughput$(\Pi, \mathcal{A}, \mathcal{E})$ over all the possible executions.*

Definition 10. *The* goodput *of protocol Π is defined to be the throughput of Π in the presence of an adversary that follows the instructions of the protocol, i.e., the processes controlled by the adversary behave honestly.*

Throughput, as defined above, should be considered as a metric of the average yield of the capacity in the worst adversarial case. *Goodput* should instead be understood as a metric of the capacity of the protocol without an adversary interfering in the protocol.

Definition 11. *Given a block-based atomic broadcast protocol Π, an adversary $\mathcal{A}$, an execution $\mathcal{E}$, and a transaction tx, we define* latency *of tx in the presence of adversary $\mathcal{A}$ in execution $\mathcal{E}$ as the number of rounds since tx is* bab-broadcast *until the first block containing tx is* bab-delivered*, and we denote it by latency$(\Pi, \mathcal{A}, \mathcal{E}, tx)$. We define the* latency of Π *to be the average number of rounds over the transactions tx in execution $\mathcal{E}$, since tx is* bab-broadcast *until the first block containing tx is* bab-delivered *and denote it by latency$(\Pi, \mathcal{A}, \mathcal{E})$.*

Definition 12. *Given a block-based atomic broadcast protocol Π, The* latency *of protocol Π is defined as* $\mathit{latency}(\Pi) = \sup_{\mathcal{A}} \mathrm{E}[\mathit{latency}(\Pi, \mathcal{A}, \mathcal{E})]$*, i.e., the supremum over all the possible adversaries $\mathcal{A}$ of the average over the randomness of the protocol of the* $\mathit{latency}(\Pi, \mathcal{A}, \mathcal{E})$ *over the possible executions $\mathcal{E}$.*

Latency is the other traditional measure of performance of the protocol. Intuitively, latency is a metric for the response time of the protocol, reflecting the amount of time needed until an operation is executed.

5 The throughput closure

We introduce a novel construction designed to enhance a given DAG protocol Π. This construction produces a DAG protocol, which we call *the throughput closure of Π* and denote by Π'. Protocol Π' possesses the unique property of ensuring that every honestly *bab-mined* block is eventually *bab-delivered*. The mechanism by which protocol Π' accomplishes this feat involves the incorporation of additional references to blocks. For any given block b, protocol Π' defines the set $\mathit{abandoned}(b)$ as the collection of valid blocks that will not be *Π-delivered* if b is to be *Π-delivered*. The block mining and delivery routines of the throughput closure Π' are built on top of their counterparts in Π. We recall that chain protocols are a subset of DAG protocols and the differences between chain protocols and general DAG protocols are the main interest of this paper.

Overview. As shown in Algorithm 1, when an honest process P_i *Π-mines* a block b, process P_i also *Π'-mines* the same block. However, in Π', the block b includes an additional set of references to the blocks in the set $\mathit{abandoned}(b)$.

The modified delivery routine operates as follows: when a block b would be *Π-delivered*, all valid blocks in the set $\mathit{abandoned}(b)$ are *Π'-delivered* in a fixed topological order immediately before b. This topological sort allows to order non *Π-delivered* blocks with respect to *Π-delivered* blocks deterministically according to the references included in the *Π-delivered* blocks. This is a crucial aspect as establishing a total order in a DAG can be generally challenging due to different processes having different partial views of the DAG. The topological sort τ ensure that all processes that have received block b agree on the same order. A canonical example for *topological sort* τ is to order the blocks in $\mathit{abandoned}(b)$ according to their *depth* in the DAG, distance to genesis, breaking the ties according to the hash of the block. Note that if an adversary creates a block with low depth, it will be only *Π-delivered* when deeper block references it, thus the adversarial block is *Π'-delivered* concurrently with deeper blocks.

Constructing the set $\mathit{abandoned}(b)$, even when it can be computed, may be challenging task, as we explained above. However, given a chain protocol Π the set $\mathit{abandoned}(b)$ becomes trivial to compute as it is formed by every block that is not an ancestor of b. Furthermore, the set $\mathit{abandoned}(b)$ is the set of leaves of the DAG, with the exception of b. As an illustrative example, Figure 2 shows the application of this construction within the context of Bitcoin. If we consider

Π to be GHOST protocol [21], we recreate the Conflux protocol [13]. Including references to the leaves in the DAG is the correct method for referring to the set $abandoned(b)$, as this set is formed by the leaves in the DAG when considering a chain protocol Π. The same approach can be considered with DAG protocols. This approach is computationally feasible even in scenarios in which the set $abandoned(b)$ may not be computable, however, some blocks may be referenced when there is no need, adding redundancy of references. Further insights into this alternative approach are provided below.

Algorithm 1 Protocol Π' for process P_i.

Implements: block-based atomic broadcast Π'
Uses: block-based atomic broadcast Π
topological sort τ

State:
$\mathcal{D}' \leftarrow \emptyset$
$b'_\ell \leftarrow [\,]$

upon event Π'-$broadcast(tx)$ **do**
  invoke Π-$broadcast(tx)$

upon event Π-$mined(b, P_j)$ **do**
  if $P_i = P_j$ **then**
    $weak \leftarrow leaves(abandoned(b, \mathcal{D}'))$
    $b' \leftarrow b$
    $b'.wrefs \leftarrow weak$
    $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{b'\}$
    invoke Π'-$mined(b', P_i)$

upon event Π'-$mined(b', P_j)$ **do**
  if $VB'(b')$ **then**
    $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{b'\}$

upon event Π-$deliver(b)$ **do**
  $ready \leftarrow ancestors'(b') \setminus ancestors'(b'_\ell)$
  $b'_\ell \leftarrow b'$
  for $b^* \in \tau(ready)$ **do**
    invoke Π'-$deliver(b^*)$

function $abandoned(b, \mathcal{D}')$ **:**
  return $\{b' \in \mathcal{D}' : b' \notin ancestors'(b) \wedge incompatible(b, b')\}$

function $VB'(b')$ **:**
  return $VB(b) \wedge \exists\ tx \in b' : undelivered(tx)$

Algorithm 2 Greedy approach for process P_i.

24: **upon event** Π-*mined*(b, P_j) **do** // Greedy approach
25: **if** $P_i = P_j$ **then**
26: $b' \leftarrow b$
27: $weak \leftarrow leaves(\{b' \in \mathcal{D}' : b' \notin ancestors(b')\})$
28: $b'.refs \leftarrow b'.refs || weak$
29: $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{b'\}$
30: **invoke** Π'-*mined*(b', P_i)

Detailed description. We describe the execution of the protocol from the perspective of an honest process P_i. When honest process P_i Π'-*broadcasts* a transaction tx, it invokes Π-*broadcast*(tx) (L3–4). Notably, the broadcast of transactions occurs exactly as it does in protocol Π. When P_i triggers event Π-*mined*(b, P_i) (L5–11), it initially computes the set *abandoned*(b) locally. To Π'-*mine* a new block b', P_i augments b by adding extra references to the set *abandoned*(b) (L7–9). Subsequently, P_i adds b' to the set of mined blocks $\mathcal{D}'$ (L10) and triggers the event Π'-*mined*(b', P_i) (L11).

When event Π'-*mined*(b', P_j) is triggered, P_i verifies the Π'-validity of b' and incorporates it into its local view (L12–14). So far, the execution of Π' closely parallels that of Π. However, the key distinction lies in the delivery of blocks (L15–19). When event Π-*deliver*(b) occurs, P_i searches for the block b' associated with b. P_i then assembles the set *ready*, which comprises the blocks to be Π'-*delivered* (L16). This set is computed as the set-difference between the ancestors of block b' and the ancestors of the last delivered block b'_l. P_i subsequently updates the last delivered block to be b' (L17). Finally, P_i applies a topological sorting algorithm τ to the set *ready* and Π'-delivers them accordingly (L18–19).

A block b' is deemed valid (L22–23) within protocol Π' if it satisfies two conditions: firstly, its associated block b must be Π-*valid*, and secondly, it must contain at least one Π'-valid transaction. Algorithm 2 presents a greedy version of *abandoned*(b). In this approach, a process P_i adds references to b' for every block that is not already an ancestor of b within protocol Π. This greedy approach solves the computational complexity of determining the set *abandoned*(b) at the expense of an excess of references.

The throughput closure mirrors protocol Π when the set *abandoned*(b) is empty for every block, indicating that the protocol does not permit the existence of abandoned blocks. However, if Π permits abandoned blocks, then there exists some executions of Π with a block b such that *abandoned*$(b) \neq \emptyset$, and the throughput closure diverges from the original protocol. The implementation of the throughput closure does entail an increase in local computation for processes. Specifically, processes need to scan the DAG and append a set of references to all leaves in *abandoned*(b) to the currently mined block b. The computational complexity of determining *abandoned*(b) can vary depending on the protocol, as discussed earlier. However, in the case of chain protocols, this set is relatively

straightforward to compute. A process simply adds references to every leaf of a chain that has not been referenced by an ancestor.

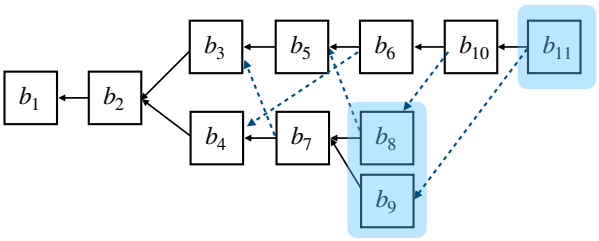


Fig. 2. An example of our construction applied to Nakamoto consensus. The full lines denote the references of the Nakamoto consensus and the blue dashed lines denote the extra references included by the throughput closure. According to Nakamoto consensus, the main chain is the chain $b_1 \cdots b_{11}$ and blocks b_4, b_7, b_8, and b_9 are abandoned. Looking at b_{11}, the set $abandoned(b_{11})$ is formed by block b_9. Blocks b_4, b_7, and b_8 are not part of the $abandoned(b_{11})$ because b_{10} already references them. When delivering b_{11}, block b_9 would be delivered between b_{10} and b_{11}.

6 Analysis

6.1 Security analysis

Theorem 1. *Given protocol DAG protocol Π implementing block-based atomic broadcast, its throughput closure Π' also implements block-based atomic broadcast.*

Proof. We demonstrate that the throughput closure Π' implements block-based atomic broadcast by leveraging the fact that Π does. Throughout this proof, we assume the perspective of an honest process P_i.

Validity: Assume that an honest process P_j *Π'-broadcasts* a given transaction tx. By construction, process P_j does so by invoking *Π-broadcast* transaction tx (L3–4). The validity property of protocol Π guarantees that process P_i eventually *Π-delivers* a block b containing transaction tx. Process P_i, by definition of the protocol, *Π'-delivers* the block b' consisting of block b with

the addition of the extra set of references (L15–19). If every transaction contained in b' is invalid, the block is not Π'-*deliver*. In the case of block b', the validity check can only fail if transaction tx fails the validity predicate. Since tx is Π'-*broadcast*, the external validity predicate is satisfied unless some block containing tx has been Π'*delivered*.
We conclude that for any honestly Π'-*broadcast* transaction tx, P_i eventually Π'-*delivers* a block b' containing tx, thus validity property of protocol Π' is satisfied.

Integrity: Process P_i only Π'-*delivers* blocks that it Π'-*delivers* or ancestors of those contained in the set $\mathcal{D}$ (L15–19). A block b' enters the set $\mathcal{D}$ only after an invokation of Π-*mined*(b, P_j). We conclude that every Π'-*delivers* has previously been Π'-*mined*.

Agreement: Consider a block b' that is Π'-*delivered* by process P_i. We consider two different cases: when b whether b is Π-*delivered* or not. On the one hand, if b is Π-*delivered* by process P_i, every honest process eventually Π-*delivers* b, thus Π'-*delivers* b' as a consequence (L15–19). On the other hand, if block bl' is Π'-*delivered* as a consequence of another block b^* is Π'-*delivered*. The same reasoning as above applies to block b^*, which implies the eventual Π'-*delivery* of block b'.

Total order: Consider two Π'*mined* blocks b'_1 and b'_2 and two honest processes P_i and P_j that $\Pi' -$ *deliver* both blocks. We distinguish four cases depending on whether blocks b_1 and b_2 are Π-*delivered* or not.
Assume that both b_1 and b_2 are Π-*delivered*. Note that in the view of any honest process the order in which blocks b_1 and b_2 are Π-*delivered* is the same as blocks b'_1 and b'_2 are Π'-*delivered* (L15–19). Due to the total order property of protocol Π, process P_i Π-*delivers* block b_1 and b_2 in the same order as process P_j, thus both processes Π'-*deliver* blocks b_1 and b_2.
If either b'_1 or b'_2 are Π'-*delivered* as a consequence of another block b_3 being Π-*delivered*. Since the set of blocks that are Π'-*delivered* as consequence of block b'_3 are Π'-*delivered* immediately before b'_3, any block b' Π'-*delivered* before (after) b' is also Π'-*delivered* before (after) the set of blocks Π'-*delivered* as a consequence of b'. The same reasoning as above applies to this case. We conclude that P_i also Π'*delivers* both b'_1 or b'_2 in the same order as P_j.
The only case left is when both b'_1 and b'_2 are Π'-*delivered* as a consequence of two blocks b'_3 and b'_4 being Π'-*delivered*. If b'_3 and b'_4 are different the case is the same as before. If b'_3 and b'_4, both P_i and P_j use the topological order to determine in which order to Π'-*delivered*. Since the topological sorting is deterministic and depends only on block b'_3, both P_i and P_j Π'-*deliver* b'_1 and b'_2 in the same order.

External validity: The external validity property is imposed by lines L22–23.

6.2 Throughput and latency

Theorem 1 states that the throughput closure Π' maintains the safety and liveness properties the original protocol Π. In this section, we delve into a comparative analysis of the performance aspects, through throughput and latency,

between Π' and Π. It is important to note that both throughput and latency definitions take into account adversarial behavior, and the connection between the adversarial behavior of Π' and Π is discussed in the following remark.

Remark 2. Note that given an adversary $\mathcal{A}'$ for protocol Π', an adversary $\mathcal{A}$ for protocol Π can be constructed by merely removing the extra references from any block that $\mathcal{A}'$ Π'-*mines*. Additionally, given an adversary $\mathcal{A}$ for protocol Π, it can also be regarded as an adversary for protocol Π', as every action taken by $\mathcal{A}$ in protocol Π is allowed in protocol Π'.

Definition 13. *Given an execution $\mathcal{E}'$ and an adversary $\mathcal{A}'$ for protocol Π', we define the equivalent execution of protocol Π as the execution $\mathcal{E}'$ without the extra references in each block and adversary $\mathcal{A}$, as discussed in Remark 2.*

Lemma 2. *Given a DAG protocol Π, its throughput closure Π' achieves the same or lower latency as Π.*

Proof. Consider an execution $\mathcal{E}'$, an adversary $\mathcal{A}'$ for protocol Π', and a transaction tx that has not already been Π'-*delivered*. Denote by $\mathcal{E}$ the equivalent execution (Definition 13) of protocol Π. Note that by definition of Π', tx has not been Π-*delivered* either (L15). Protocol Π' has two different mechanisms to Π'-*deliver*(tx).

On the one hand, if an event Π-*deliver*(b) for a block b containing tx is triggered, then b is Π'-*delivered* (L15). In this case, $\mathit{latency}(\Pi', \mathcal{A}', \mathcal{E}', tx)$ is the same as $\mathit{latency}(\Pi, \mathcal{A}, \mathcal{E}, tx)$.

On the other hand, if an event Π-*deliver*(b') for a block b' that does not contains tx but is descendent of a block b containing tx., then block b' is Π'-*delivered* immediately before b (L16). In this case, $\mathit{latency}(\Pi', \mathcal{A}', \mathcal{E}', tx)$ is strictly smaller than $\mathit{latency}(\Pi, \mathcal{A}, \mathcal{E}, tx)$.

We conclude that for every adversary, execution, and transaction, the latency of protocol $\mathit{latency}(\Pi', \mathcal{A}', \mathcal{E}', tx) \leq \mathit{latency}(\Pi, \mathcal{A}, \mathcal{E}, tx)$. Hence, $\mathit{latency}(\Pi') \leq \mathit{latency}(\Pi)$

The next result clarifies the motivation for the term throughput closure.

Lemma 3. *Given a DAG protocol Π, then $\mathit{throughput}(\Pi') \geq \mathit{throughput}(\Pi)$, and if Π permits abandoned blocks, then $\mathit{throughput}(\Pi') > \mathit{throughput}(\Pi)$.*

Proof. Consider an execution $\mathcal{E}'$, an adversary $\mathcal{A}'$ for protocol Π', and a transaction tx that has not already been Π'-*delivered*. Denote by $\mathcal{E}$ the equivalent execution (Definition 13) of protocol Π.

On the one hand, if there is no abandoned block in the execution $\mathcal{E}'$, then, the set $\mathit{abandoned}(b')$ is empty for every block b'. Thus, no extra reference is added at any point in the execution of Π' the executions $\mathcal{E}$ and $\mathcal{E}'$ identical. We conclude that $\mathit{throughput}(\Pi, \mathcal{A}', \mathcal{E}) = \mathit{throughput}(\Pi', \mathcal{A}, \mathcal{E})$.

On the other hand, if there exists at least one abandoned block b' in execution $\mathcal{E}'$, then, the set $\mathit{abandoned}(b^*)$ is not empty for some block b^* that is eventually Π'-*delivered*. When b^* is Π'*delivered* so is b' (L16).

We conclude that $\mathit{throughput}(\Pi', \mathcal{A}', \mathcal{E}') \geq \mathit{throughput}(\Pi', \mathcal{A}, \mathcal{E})$ for every possible adversary $\mathcal{A}'$ and execution $\mathcal{E}'$, thus $\mathit{throughput}(\Pi') \geq \mathit{throughput}(\Pi')$. Furthermore, if Π permits abandoned blocks, there exists an λ-partial execution with a block b that is abandoned in all its extensions. This means that the probability, over the randomness of the protocol, of having an abandoned block is strictly greater than zero (Remark 1). Thus, $\mathrm{E}[\mathit{throughput}(\Pi', \mathcal{A}', \mathcal{E}')] > \mathrm{E}[\mathit{throughput}(\Pi, \mathcal{A}, \mathcal{E})]$ for at least some adversary $\mathcal{A}'$. We conclude by noticing that if an adversary $\mathcal{A}^*$ prevents the exclusion of abandoned blocks, then $\mathit{throughput}(\Pi', \mathcal{A}^*, \mathcal{E}') > \mathit{throughput}(\Pi', \mathcal{A}', \mathcal{E}')$. Hence, we conclude that

$$\begin{aligned}\mathit{throughput}(\Pi') &= \inf_{\mathcal{A}'} \mathrm{E}[\mathit{throughput}(\Pi', \mathcal{A}', \mathcal{E}')] \\ &> \inf_{\mathcal{A}} \mathrm{E}[\mathit{throughput}(\Pi, \mathcal{A}, \mathcal{E})] = \mathit{throughput}(\Pi).\end{aligned}$$

Corollary 1. *Given a DAG protocol Π, then $\mathit{goodput}(\Pi') \geq \mathit{goodput}(\Pi)$. Furthermore, if Π allows for the existence of abandoned blocks, then $\mathit{goodput}(\Pi') > \mathit{goodput}(\Pi)$.*

Proof. Consider the proof of Lemma 3 limited to adversaries that follow the instructions of the protocol.

Note that every chain protocol trivially permits abandoned block. We can finally conclude that DAG protocols are strictly better then chain protocols.

Theorem 2. *Given a chain protocol Π, there exists a DAG protocol Π' such that: $\mathit{latency}(\Pi') \leq \mathit{latency}(\Pi)$ and $\mathit{throughput}(\Pi') > \mathit{throughput}(\Pi)$.*

Proof. Lemma 1 states that a chain protocol Π permits abandoned blocks. Theorem 1 demonstrates that its throughput closure Π' implements block-based atomic broadcast. Lemma 3 shows that $\mathit{throughput}(\Pi') > \mathit{throughput}(\Pi)$. Finally, Lemma 2 establishes that $\mathit{latency}(\Pi') \leq \mathit{latency}(\Pi)$.

References

1. Chia network. `https://docs.chia.net/docs/01introduction/what-is-chia`
2. Alpos, O., Amores-Sesar, I., Cachin, C., Yeo, M.: Eating sandwiches: Modular and lightweight elimination of transaction reordering attacks. CoRR **abs/2307.02954** (2023)
3. Amores-Sesar, I., Cachin, C., Tedeschi, E.: When is spring coming? A security analysis of avalanche consensus. In: OPODIS. LIPIcs, vol. 253, pp. 10:1–10:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022)
4. Bagaria, V.K., Kannan, S., Tse, D., Fanti, G., Viswanath, P.: Prism: Deconstructing the blockchain to approach physical limits. In: CCS. pp. 585–602. ACM (2019)
5. Bowman, M., Das, D., Mandal, A., Montgomery, H.: On elapsed time consensus protocols. In: INDOCRYPT. Lecture Notes in Computer Science, vol. 13143, pp. 559–583. Springer (2021)
6. Cachin, C., Guerraoui, R., Rodrigues, L.E.T.: Introduction to Reliable and Secure Distributed Programming (2. ed.). Springer (2011)

7. Castro, M., Liskov, B.: Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. **20**(4), 398–461 (2002)
8. Danezis, G., Kokoris-Kogias, L., Sonnino, A., Spiegelman, A.: Narwhal and tusk: a dag-based mempool and efficient BFT consensus. In: EuroSys. pp. 34–50. ACM (2022)
9. David, B., Gazi, P., Kiayias, A., Russell, A.: Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 10821, pp. 66–98. Springer (2018)
10. Keidar, I., Kokoris-Kogias, E., Naor, O., Spiegelman, A.: All you need is DAG. In: PODC. pp. 165–175. ACM (2021)
11. Keidar, I., Naor, O., Poupko, O., Shapiro, E.: Cordial miners: Fast and efficient consensus for every eventuality. In: DISC. LIPIcs, vol. 281, pp. 26:1–26:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023)
12. Lamport, L.: The part-time parliament. ACM Trans. Comput. Syst. **16**(2), 133–169 (1998)
13. Li, C., Li, P., Zhou, D., Yang, Z., Wu, M., Yang, G., Xu, W., Long, F., Yao, A.C.: A decentralized blockchain with high throughput and fast confirmation. In: USENIX Annual Technical Conference. pp. 515–528. USENIX Association (2020)
14. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Whitepaper (2009), `http://bitcoin.org/bitcoin.pdf`
15. Natoli, C., Gramoli, V.: The balance attack or why forkable blockchains are ill-suited for consortium. In: DSN. pp. 579–590. IEEE Computer Society (2017)
16. Pedone, F., Schiper, A.: Generic broadcast. In: DISC. Lecture Notes in Computer Science, vol. 1693, pp. 94–108. Springer (1999)
17. Penzkofer, A., Kusmierz, B., Capossele, A., Sanders, W., Saa, O.: Parasite chain detection in the IOTA protocol. CoRR **abs/2004.13409** (2020)
18. Popov, S., Saa, O., Finardi, P.: Equilibria in the tangle. Comput. Ind. Eng. **136**, 160–172 (2019)
19. Rocket, T., Yin, M., Sekniqi, K., van Renesse, R., Sirer, E.G.: Scalable and probabilistic leaderless BFT consensus through metastability. CoRR **abs/1906.08936** (2019)
20. Sompolinsky, Y., Wyborski, S., Zohar, A.: PHANTOM GHOSTDAG: a scalable generalization of nakamoto consensus: September 2, 2021. In: AFT. pp. 57–70. ACM (2021)
21. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in bitcoin. In: Financial Cryptography. Lecture Notes in Computer Science, vol. 8975, pp. 507–527. Springer (2015)
22. Spiegelman, A., Giridharan, N., Sonnino, A., Kokoris-Kogias, L.: Bullshark: DAG BFT protocols made practical. In: CCS. pp. 2705–2718. ACM (2022)

Assessing the Impact of Sanctions in the Crypto Ecosystem: Effective Measures or Ineffective Deterrents?

Francesco Zola[1 [0000−0002−1733−5515]], Jon Ander Medina[1 [0009−0008−1107−0617]], and Raúl Orduna[1 [0000−0002−5932−0987]]

Vicomtech Foundation, Basque Research and Technology Alliance (BRTA);
Paseo Mikeletegi, 57, Donostia 20009, Spain
{fzola, jmedina, rorduna}@vicomtech.org

Abstract. Regulatory authorities aim to tackle illegal activities by targeting the economic incentives that drive such behaviour. This is typically achieved through the implementation of financial sanctions against the entities involved in the crimes. However, the rise of cryptocurrencies has presented new challenges, allowing entities to evade these sanctions and continue criminal operations. Consequently, enforcement measures have been expanded to include crypto assets information of sanctioned entities. Yet, due to the nature of the crypto ecosystem, blocking or freezing these digital assets is harder and, in some cases, such as with Bitcoin, unfeasible. Therefore, sanctions serve merely as deterrents. For this reason, in this study, we aim to assess the impact of these sanctions on entities' crypto activities, particularly those related to the Bitcoin ecosystem. Our objective is to shed light on the validity and effectiveness (or lack thereof) of such countermeasures. Specifically, we analyse the transactions and the amount of USD moved by punished entities that possess crypto addresses after being sanctioned by the authority agency. Results indicate that while sanctions have been effective for half of the examined entities, the others continue to move funds through sanctioned addresses. Furthermore, punished entities demonstrate a preference for utilising rapid exchange services to convert their funds, rather than employing dedicated money laundering services. To the best of our knowledge, this study offers valuable insights into how entities use crypto assets to circumvent sanctions.

Keywords: Sanctions circumvention, Money laundering, Flow analysis, Behavioural analysis, Cryptocurrency, Traceability

1 Introduction

Understanding the multidimensional nature of crime is crucial for developing effective strategies to prevent and combat these illicit activities. Crimes often manifest in various forms and domains, like drug trafficking, human trafficking, and other types of organised crime. Nevertheless, in all these cases, the main

goal of the criminal networks is still to make a profit [23]. For this reason, disrupting the economic incentives driving illicit behaviour has become the primary objective in tackling these crimes [29]. This task requires cooperation and coordination among governments, law enforcement agencies, financial institutions, regulatory bodies, and other stakeholders at national and international levels.

This is the case of authority agencies such as the Office of Foreign Assets Control (OFAC)[1], Office of Financial Sanctions Implementation (OFSI)[2], European External Action Service (EEAS)[3] and United Nations Security Council (UNSC)[4], that aim to implement and enforce financial sanctions directly to the entity behind some violations and crimes. In fact, these agencies have the authority to freeze, block or restrict access to sanctioned entities' assets such as banking accounts, real estate, vessels, etc.

However, with the advent of virtual currencies such as cryptocurrencies, stablecoins, and Non-Fungible Tokens (NFTs), sanctioned entities have discovered new opportunities for circumventing sanctions and continuing their illicit activities [30]. These digital assets promote decentralization and offer varying degrees of anonymity or pseudo-anonymity, creating a borderless ecosystem ideal for the proliferation of illicit activities [14][25]. According to "*The 2024 Crypto Crime Report*" [12], although illicit crypto-transactions constitute less than 0.5% of the total on-chain transaction volume, they accounted for nearly 40 billion USD in 2022 and over 24 billion USD in 2023.

The significant size of the crypto market, the opportunities crypto assets present, and their proven involvement in illicit activities [9][16] have raised concerns about ensuring compliance with existing financial regulations. While some countries such as Tunisia, Nepal, Libya, Iraq, Bolivia, and Algeria have banned the use of cryptocurrency [7], others have directed their effort to implement new frameworks and technology to increase their control degree over the crypto ecosystem. Thus, regulatory agencies have intensified their enforcement actions against sanctioned entities, including tracking information about their crypto assets whenever possible. However, due to the nature of the crypto ecosystem, they still face limitations in blocking these digital assets. As a result, sanctions solely serve as a deterrent, aiming to discourage other individuals and companies from engaging in transactions with punished entities. Yet, this limitation makes crypto assets the perfect facilitator for ongoing illicit operations and circumventing sanctions.

For this reason, in this work, we aim to examine the impact these sanctions generate in the crypto activities of punished entities and their related violations. Our objective is to evaluate the validity and effectiveness (or lack thereof) of such countermeasures. Specifically, we analyse the transactions and the amount of USD moved by punished entities that possess crypto addresses before and after being sanctioned by the authority agency. Furthermore, we investigate which

[1] https://ofac.treasury.gov/
[2] https://sanctionssearchapp.ofsi.hmtreasury.gov.uk/
[3] https://www.eeas.europa.eu/
[4] https://www.un.org/securitycouncil/content/un-sc-consolidated-list

type of known crypto entities (Exchange, Mixers, Gambling, etc.) are typically engaged with by the punished entities post-sanction. Thus, we investigate if they are attempting to connect with other entities involved in illicit operations or are employing strategies such as money laundering, on-ramps and off-ramps operations, funding raise campaigns, etc.

More specifically, this work analyses entities (individuals and companies) sanctioned by the OFAC agency that have information about Bitcoin (BTC) assets. These decisions were taken for two specific reasons: a) Bitcoin (together with stablecoins) is widely recognised among the most popular cryptocurrencies for illicit activities [12], primarily due to its high market value and accessibility, even for users without technical background; b) other authority agencies (OFSI, EEAS, UNSC) do not have a comprehensive list of sanctioned crypto-related entities or do not release it publicly.

The results suggest that sanctions have been effective for roughly half of the sanctioned entities, while the others continue to engage in transactions through sanctioned Bitcoin addresses. Additionally, sanctioned entities prefer to directly convert their cryptocurrencies using dedicated services (*Exchanges*), rather than apply money laundering strategies using services like *Mixers* or *Gambling*.

To the best of our knowledge, this study offers a first step towards determining the effectiveness of sanctions within the crypto ecosystem and how sanctioned entities used them to circumvent sanctions.

2 Background

This section presents an overview of the crypto ecosystem, showing regulations, directives, and literature approaches. Specifically, in Section 2.1 a review of European Union regulations related to the crypto ecosystem is presented, while Section 2.2 is focused on presenting the United States OFAC agency and its crypto sanctions. Finally, Section 2.3 describes related research in the field.

2.1 EU Directive review

As mentioned, crypto assets can be harder (or in some cases unfeasible) to freeze or block directly. However, governments and regulatory authorities worldwide have been increasingly implementing measures to monitor and regulate crypto markets to tackle concerns related to illicit activities. Thus, these regulations aim to facilitate the imposition of sanctions when necessary.

The European Union (EU) has established and reviewed several directives aimed at tackling issues such as money laundering, fraud, terrorism financing, and other emerging challenges related to non-cash payments. For instance, in 2015, the 5th Anti-Money Laundering Directive (5AMLD) [1] was introduced to address new trends in terrorist financing, building upon the provisions of the 4AMLD. Notably, under the 5AMLD, the role of cryptocurrency Exchanges has changed, they are now considered equivalent to financial institutions. Therefore, they are required, among other measures, to adhere to Know Your Cus-

tomer (KYC) requirements, implement Anti-Money Laundering (AML) mechanisms, and register with national regulatory authorities. This directive has been amended to include provisions regarding information accompanying transfers of funds and certain crypto-assets, through the Regulation (EU) 2023/1113 [6]. The amendment fosters international cooperation within the Financial Action Task Force (FATF) and the global implementation of its recommendations [4]. Furthermore, it sets the obligation for virtual asset service providers (VASPs) and crypto asset service providers (CASPs) to collect information about the person who uses their services. Together with these directives, another pivotal regulation is the Markets in Crypto-Assets (MiCA) [5], which clearly distinguishes between different types of crypto-assets (asset-referenced tokens, electronic money or e-money, and other crypto assets). It then imposes constraints on CASPs to ensure market integrity and financial stability. One of its key provisions involves significant disclosure and transparency rules aimed at better informing consumers about associated risks, as well as mandating the implementation of security measures and anti-money laundering compliance.

Fraud and counterfeiting of non-cash means of payment is regulated through the EU directive 2019/713 [3]. The framework establishes measures aimed at preventing and detecting fraud and counterfeiting of non-cash payment instruments, such as security requirements for payment service providers, customer authentication procedures, and usage of secure technologies (encryption and tokenization). This directive aimed to cover also new types of non-cash payment instruments such as e-money and virtual currencies, since they have a significant cross-border dimension. Another interesting EU framework, although it doesn't specifically mention digital currency or cryptocurrencies due to its general aim, is the Directive (EU) 2017/1371 [2] that defines legal framework and measures to combat any fraud against the financial interests of EU.

Despite the revision, update, and implementation of these policies, freezing or blocking crypto assets remains harder to accomplish by technical design. Its boundary-less structure, the availability of services in jurisdictions where these policies don't apply, or in countries unwilling to cooperate in criminal investigations, the anonymity of users, and the ease of conducting transactions - these properties collectively enable users to circumvent sanctions and persist in their illicit activities.

2.2 US Office of Foreign Assets Control

The Office of Foreign Assets Control (OFAC) is part of the Department of the United States (US) Treasury, and it is in charge of implementing economic and trade sanctions in accordance with US foreign policy and national security objectives. These sanctions are directed towards specific foreign countries and regimes, terrorists, international narcotics traffickers, individuals involved in the proliferation of weapons of mass destruction, and other actors posing threats to the national security, foreign policy, or economy of the United States. These actors and their blocked assets are included in a *Specially Designated Nationals and*

Blocked Persons (SDN) list[5]. Consequently, US individuals and companies are generally prohibited from dealing with them. The actors reported in the SDN list, which we refer to as *entities* in this paper, are sanctioned due to the violation of one (or more) Executive Orders and/or Code of Federal Regulations (CFR)[6]. Since 2018, OFAC has included cryptocurrency-related information in the SDN list as blocked assets for sanctioned entities whenever such information is available. In some cases, one entity can have multiple sanctioned addresses. To date, the violations that have led to sanctions against entities with crypto addresses involved, are detailed in Table 1. In this paper

#	Code	Description	Executive Order N.
1	CYBER2	Malicious Cyber Activities	13694, 13757
2	DPKR3	Blocking Property of the Government of North Korea	13722
3	DPKR4	Additional Sanctions With Respect to North Korea	13810
4	ELECTION	Foreign Interference in the US Election	13848
5	IFSR	Iranian Financial Sanctions Regulations	31 CFR part 561
6	ILLICTI-DRUGS	Illicit Drug Trade	14059
7	IRGC	Iranian Financial Sanctions	31 CFR Part 561
8	NPWMD	Weapons of Mass Destruction Proliferators Sanctions	31 CFR part 544
9	RUSSIA	Blocking Harmful Activities of the Russian Federation	14024
10	SDGT	Narcotics Trafficking Sanctions	31 CFR part 594
11	SDNTK	Foreign Narcotics Kingpin Sanctions	31 CFR part 598

Table 1. Violations reported in the SDN list that have generated sanctions against cryptocurrency related entities.

2.3 Cryptocurrency and Cybercrime

Cryptocurrencies have created a convenient ecosystem for the permanence and movement of illicit cybercrime-related activities, with currencies such as Bitcoin, Ethereum, and Monero becoming their operational space. The evolution of illicit activities within the cryptocurrency ecosystem has been studied extensively [13] [11]. For instance, Hornuf et al. [17] analysed cybercrime related to Ethereum transactions. In particular, they identified over 1.78 million transactions related to 19 categories of cybercrime, with losses amounting to $1.65 billion up to the year 2021, posing the focus on estimating how these cybercrimes impact victims' risk-taking, risk-adjusted returns, and investor behaviour. In the same line, in [26], authors employ the Generalized Autoregressive Score (GAS) model to examine the impact of cybercrime on cryptocurrency returns in South Africa. On the other hand, in [10], authors attempted to correlate the expansion of ransomware activities with transactions performed in these cryptocurrencies between 2015 and 2020. However, although it is clear that these cryptocurrencies

[5] https://sanctionssearch.ofac.treas.gov/

[6] https://ofac.treasury.gov/specially-designated-nationals-list-sdn-list/program-tag-definitions-for-ofac-sanctions-lists

facilitate the growth of ransomware revenue [12], authors did not find an evident correlation. In [8], traditional machine learning algorithms are used to detect illegal activities using a Bitcoin dataset, while in [19] graph-based networks are used for a similar task. Similar applications are also explored in [20] for defining a method to identify and trace illicit activities in the Ethereum blockchain.

Among the most relevant cybercrimes, cryptocurrencies have become the perfect solution for laundering illegal funds [27]. By fostering decentralization, user anonymity, and the ease of making cross-border transactions, they have led to the proliferation of dedicated services. Achraf Guidara [16] examines the relationship between cryptocurrencies and money laundering, emphasising the urgent need to develop a robust and internationally coordinated regulatory framework to mitigate these risks. In [21], 182 Bitcoin addresses belonging to 56 members of the Conti ransomware group are analysed with the aim of identifying if money laundering mechanisms are applied. They conclude that cryptocurrency exchanges and dark web services are involved in 71% and 30% of transactions, respectively, while only 8% utilized mixers. These findings challenge the prevailing notion that cybercriminals employ sophisticated methods, highlighting instead the simplicity of their tactics [22].

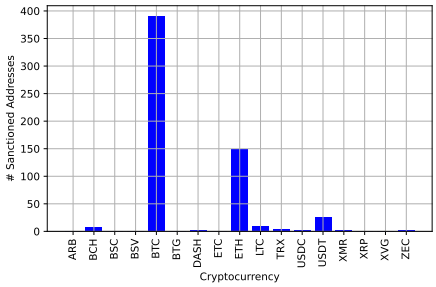
At the same time, as introduced in the previous sections, the lack of market control has turned these cryptocurrencies into a means of evading sanctions, allowing entities to continue their illicit activities. For this reason, in this work, we aim to analyse how punished entities use crypto assets to circumvent sanctions and whether they also employ laundering mechanisms to increase their anonymity.

3 Experimental Framework

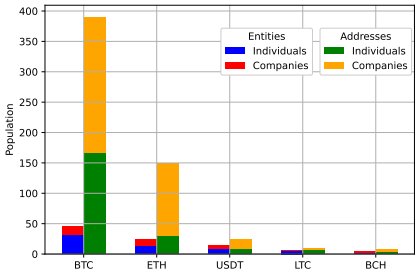
In this section, the dataset and the approach followed in this study are presented. More specifically, Section 3.1 reviews the sanctioned list used, while Section 3.2 introduces the Bitcoin dataset. Finally, the guidelines and key concepts used during the experiments are reported in Section 3.3.

3.1 Sanctions List

As of February 2024, the SDN list contains information about 600 crypto addresses related to 17 different cryptocurrencies, as shown in Figure 1a. The figure shows that the majority of reported addresses ($\sim$ 65%) belong to the Bitcoin (BTC) network, while another $\sim$ 25% are from Ethereum (ETH). The remaining addresses, representing just 10%, are divided across 15 cryptocurrencies. Furthermore, analysing the composition of the sanctioned entities (Figure 1b) that belong to the top-5 sanctioned cryptocurrencies, it becomes evident that while there are more punished individuals than companies for both BTC and ETH, companies possess a greater number of punished addresses. These results lead us to focus the analysis only on the BTC-related entities, as anticipated

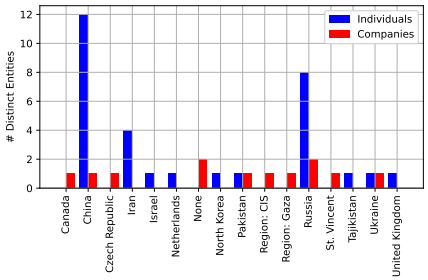


(a) Distribution of sanctioned cryptocurrencies in the SDN list.

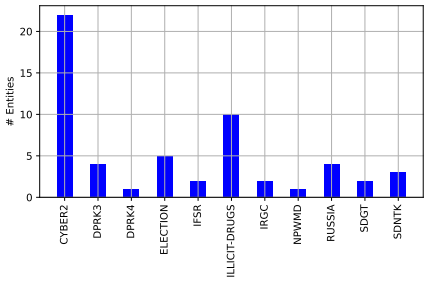


(b) Addresses and entities distribution in the SDN list (top-5 populated cryptocurrencies).

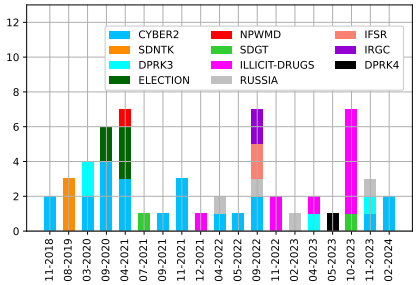
Fig. 1. Overall statistics of sanctioned entities with cryptocurrency information extracted from the SDN list (February 2024).



(a) Distribution of BTC-related entities in the SDN list per country/region.



(b) Distribution of BTC-related entities in the SDN list per violation.



(c) BTC-related entities per violation over time.

Fig. 2. Overview about violations of sanctioned entities with BTC information extracted from the SDN list (February 2024).

in Section 2.2. In particular, this constraint leaves us to study 43 out of the 56 available entities in the SDN list.

Figure 2a details that most of the BTC-related sanctioned entities (both companies and individuals) are located in China and Russia. In fact, of the 43 punished entities, 13 are from China and 10 from Russia ($\sim$ 54%). In both cases, the number of sanctioned individuals overwhelms the number of companies, while there are only sanctioned companies in Canada, the Czech Republic, St. Vincent, and in the regions of Gaza and the Commonwealth of Independent States (CIS). Figure 2b shows the distribution of these sanctioned entities with respect to their violations. It is to be noted that one entity can face sanctions for multiple violations. Consequently, the most prevalent category, with 23 entities, pertains to malicious cyber-enabled activities since they include a broad spectrum of crimes. Furthermore, among the most populated categories, 10 entities are sanctioned for illicit drug trading, while 5 entities are linked to interference in the US election. Finally, Figure 2c analyses the temporality of these sanctions and the number of entities involved. The figure shows that US authorities led a significant operation on illicit drug trading in October 2023, resulting in the sanctioning of 6 entities. Another interesting finding is that sanctioning operations against specific violations tend to occur only on specific dates. Specifically, violations related to *CYBER2*, *ILLICIT-DRUGS*, and *RUSSIA* are consistently detected over time, while the others are detected only on specific dates.

3.2 Bitcoin Dataset

In this paper, the entire Bitcoin blockchain data until the block 830,000 are downloaded, i.e., all the transactions until February 11th, 2024 (more than 900M transactions). On the other hand, to have more information about real-world entities, labelled (tagged) addresses are gathered from multiple reliable sources, such as WalletExplorer[7] and the tagpacks provided by Graphsense[8]. Indeed, these sources represent valid solutions used in many previous researches [24][28][32], and allowed us to gather more than 38M addresses of almost 400 entities labelled as *Exchanges, Gambling, Marketplaces, Mining Pools, Mixers, Services, Trading platforms, eWallet, Ransomware, Sextortion,* and *Extremist.*

3.3 Proposed Analysis

As mentioned in Section 3.1, the BTC addresses included in the SDN list represent the starting point of our investigation. More specifically, from each of them, we analyse the address-transaction graph [15][32]. This graph is directly built using the information available in the BTC blockchain, where nodes are BTC addresses and transactions. Then directed edges (arrows) from addresses to transactions represent incoming relations, while edges from transactions to addresses are outgoing relations, as shown in Figure 3. Furthermore, the edge

[7] https://www.walletexplorer.com/
[8] https://graphsense.info/

may incorporate BTC information like amount, fee, timestamps, etc. With these principles, it is possible to define the n-step address-transaction graph of a sanctioned address $X1$, as a graph in which all the paths from $X1$ involve maximum n transactions. Thus, the paths from $X1$ have a maximum length of $2n$ (Figure 3).

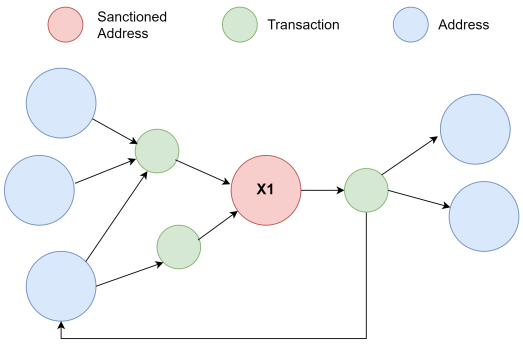


Fig. 3. An example of a *1*-step address-transaction graph

In this work, we present two different analyses: the first one is based on assessing the effectiveness of the sanctions by analysing the activities of the entities (*flow analysis*), and the second has the aim to detect the relations that entities have after being sanctioned (*behavioural analysis*).

For the *flow analysis*, a temporal aspect is introduced in the address-transaction graph. Specifically, for each address of each entity, multiple 1-step address-transaction graphs are created, considering 4 different temporal ranges: a) all the transactions prior to the imposed sanction (*pre-sanction*); b) transactions achieved immediately after the sanctions within the subsequent 7 days (*7 post-sanction*); c) transactions achieved immediately after the sanctions within 30 days post-sanctions (*30 post-sanction*); d) and finally all the activities post-sanctions up to February 11th, 2024 (*up-to-date*). These ranges allow us to evaluate the behaviour of the sanctioned entity and detect how they react to this situation in short, medium, and long terms.

Once these graphs are built, from each one, several metrics such as the number of input and output transactions, the overall balance of the entity after each temporal range, and the amount in USD of money sent and received by the entity (the BTC/USD value is fixed on the day the transaction is performed), are extracted and used for evaluating the trends and the effectiveness of the sanctions. It is to be noted that one entity may possess multiple sanctioned addresses. Therefore, metrics computed from each of its addresses are aggregated to provide a comprehensive overview of the entity's behaviour.

On the other hand, the *behavioural analysis* is based on analysing a single address-transaction graph for each entity, that is created using data from im-

mediately after the sanctions until the end of the dataset (*up-to-date*). Furthermore, this graph is enriched with real-world entity information, i.e., with labels gathered from the external sources mentioned in Section 3.2. This approach enables us to identify whether the sanctioned entity engages in transactions with other known entities, which could include potential actors related to illicit operations (e.g. other sanctioned entities, ransomware, etc.) or it tries to apply strategies for conducting activities such as money laundering (e.g. involving mainly mixers, gambling or other services), on-ramps and off-ramps operations (e.g. involving exchanges), funding raise campaign (e.g. involving mining pool or marketplace). This *behavioural analysis* is performed considering both 1-step and 2-step address-transaction graphs. This approach gives us a deeper view of the entity strategy, since the 1-step analysis only provides information about its direct relations, while the 2-step also includes undirected transactions (reached in two steps).

4 Current Study

In this section, the results obtained in the two proposed analyses are presented. In particular, Section 4.1 describes the results obtained analysing transactions and money flow of the entities before and after their sanctions, while Section 4.2 details the relations that the sanctioned entities have had with other known type of entities. Finally, discussions and limitations are reported in Section 4.3.

4.1 Flow Analysis Results

Figure 4 shows the number of entities that received and sent money through crypto transactions post-sanctions. Specifically, the figure illustrates that only half of all the sanctioned entities were effectively discouraged from engaging in transactions. In particular, only 21 entities stopped to receive money, and 25 to send funds. On the other hand, the figure reveals that despite the sanctions, some entities (7) continued to move funds within 7 days of the OFAC sanction. For this reason, Figure 5 enables us to comprehend how these funds are being moved, analysing the balance in terms of BTC held by each entity in its sanctioned addresses before and after the sanctions.

Although Figure 4 indicates that some entities were not deterred from conducting transactions, Figure 5 emphasises a general trend of maintaining at least a minimal balance in the sanctioned addresses, excluding the two entities with the highest balance ($\geq$ 50 BTC) who adopted an off-ramp strategy. Yet, the number of entities with a balance of 0 decreased, while the number of entities with a balance in a range > 0 and ≤ 0.1 BTC increased.

Table 2 reports the number of transactions that involve sanctioned entities, categorised by the violation they are convicted of. Additionally, the table includes the USD volume moved by the entities for each violation, both before and after the sanctions. In particular, the volume indicates both incoming and

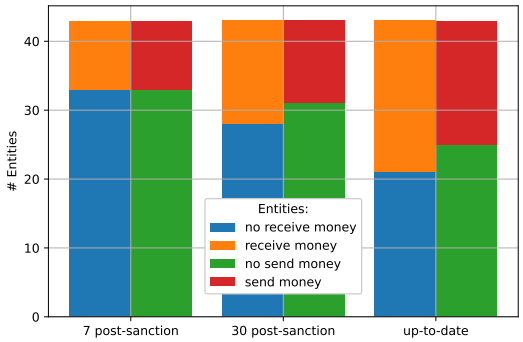


Fig. 4. Number of sanctioned entities that perform transactions (received and sent) in the different post-sanction intervals.

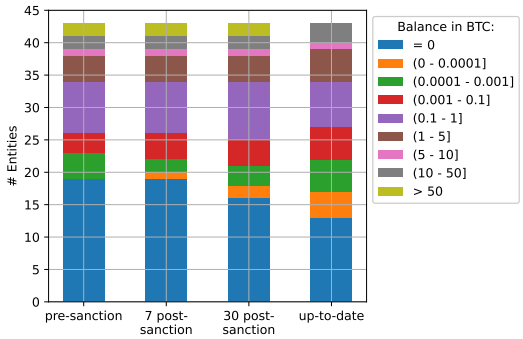


Fig. 5. Entity balance (in BTC) considering the sanctioned addresses at the four stages: *pre-sanction, 7 post-sanction, 30 post-sanction,* and *up-to-date.*

outgoing transactions. Results in Table 2 show that *CYBER2* represents the violation with the highest number of transactions pre-sanction (more than 150K) and the highest USD volume of about 8,300 million. The result is expected since this violation also has the highest number of sanctioned entities (Figure 2b). However, what is interesting, is that after the sanctions, entities related to this violation still perform 305 transactions for a market of about 3 million USD. On the other hand, entities related to *DPRK3* and *ILLICIT-DRUGS* violations perform a high number of transactions and show a USD volume of 240 million and 120 million pre-sanction, respectively. Although these numbers decrease in the post-sanction phase, they remain consistent, with 209 thousand USD for *DPRK3* and 8 million USD for *ILLICIT-DRUGS*. Also, the market related to *RUSSIA* violation is pretty high pre-sanctions, with almost 40 million USD moved in just 553 transactions. However, it seems strongly affected by the sanctions, resulting in just 4 transactions with an overall amount of 162 USD. Finally, entities related to violations such as *IFSR, IRGC* and *SDGT* are indeed deterred from performing transactions; in fact, they achieve only 1 transaction post-sanctions, moving just a few dollars (or less). Notably, the entities involved in *DPRK4* activities are shut down. More precisely, they have not achieved transactions from May 2023 (Figure 2c) until the date.

		# Transactions		USD Volume	
#	**Violation**	**Pre-Sanction**	**Up-to-date**	**Pre-Sanction**	**Up-to-date**
1	*CYBER2*	153 K	305	8,300 M	3 M
2	*DPRK3*	2 K	63	240 M	209 K
3	*DPRK4*	62	0	10 M	0
4	*ELECTION*	46 K	8	6 M	1 K
5	*IFSR*	182	1	461 K	≤ 1
6	*ILLICIT-DRUGS*	9 K	747	120 M	8 M
7	*IRGC*	182	1	461 K	≤ 1
8	*NPWMD*	97	8	26 K	1 K
9	*RUSSIA*	553	4	39 M	162
10	*SDGT*	18 K	1	42 M	103
11	*SDNTK*	354	42	104 K	12 K

Table 2. Number of transactions achieved and estimation of USD moved before and after the sanctions for each violation.

4.2 Behavioural Analysis Results

Table 3 reports the results gathered during the behavioural analysis, involving 1-step and 2-step address-transaction graphs. In particular, by creating a 1-step graph from each of the 43 entities (387 BTC-sanctioned addresses) it is possible to reach about 4K addresses, of which only 340 addresses (8.48%) are related to known entities and have an external label. On the other hand, increasing the analysis considering also undirected connections (2-step address-transaction

graphs), it is possible to reach more than 10M addresses, of which just 175,444 (1.67%) are labelled. These outcomes confirm that, although the 1-step analysis has more labelled information, it identifies only 15 known entities of 4 different behaviours, while the 2-step analysis enriches the investigation by uncovering 67 entities with 9 different behaviours.

The results reported in Table 3 highlight that, in both the 1-step and 2-step analyses, the majority of labelled addresses belonged to *Exchanges* with 97% and 86%, respectively. At the same time, in the 2-step analysis, 13.5% of the labelled addresses are linked to 11 crypto services (trading, eWallet, banking, etc.). This enhanced scenario also highlights connections between sanctioned entities and addresses associated with *Sextortion, Ransomware* and *Extremism* crimes.

		1-step analysis		**2-step analysis**	
#	**Behaviour**	**# Distinct Entities**	**# Distinct Address**	**# Distinct Entities**	**# Distinct Address**
1	*Service*	3	5	13	23,727
2	*Mixer*	1	1	2	14
3	*Exchange*	9	331	37	151,385
4	*OFAC Sanctioned*	2	3	6	33
5	*Gambling*	-	-	1	249
6	*Extremism*	-	-	1	13
7	*Ransomware*	-	-	2	9
8	*Mining Pool*	-	-	4	13
9	*Sextortion*	-	-	1	1
	Labelled	**15**	**340**	**67**	**175,444**
	No Labelled	**-**	**3,668**	**-**	**10,356,787**

Table 3. Behavioural analysis of output entities related to sanctioned actors considering 1-step and 2-step address-transaction graphs.

4.3 Discussion and Limitations

Discussion. New regulations have started to treat Exchanges as financial services, requiring any Crypto Asset Service Provider or Virtual Asset Service Provider to implement anti-money laundering control measures such as Know Your Customer (KYC) policies. Nevertheless, criminals have started to explore new methods to obtain money fraudulently and to launder and use it, e.g., including dedicated services and other digital assets that current solutions/directives do not adequately supervise. Indeed, these new money laundering methods exploit gaps in existing legislations, which in turn require frequent updates and make them challenging to enforce and follow. In this context, it is crucial to consider and incorporate also technical connectivity to the crypto ecosystem, i.e., ensuring the technology used by both users and service providers is connected and operates under unified legal standards. This alignment will help enforce the

law and catch those who commit crimes, making it more difficult for criminals to exploit any legal loopholes.

Aligned with the findings presented in [27], this study, using OFAC information, shows that Bitcoin is among the most used cryptocurrencies for various crimes, not limited to the cyber ecosystem. The results indicate that, despite being sanctioned, entities still perform operations with their blocked or frozen cryptocurrencies without employing complex tactics or dedicated money laundering services in their transactions. In fact, entities prefer to achieve off-ramp activities through known and reliable exchanges. These results are aligned with the outcomes reported in [21] and [22] regarding ransomware funds. However, the results also show that, in some cases, sanctioned entities maintain relationships with other sanctioned entities or entities involved in other crimes, such as sextortion, ransomware, and extremism.

This paper shows that a 1-step analysis is not sufficient for understanding and tracing the operations of sanctioned entities. Indeed, the analysis reveals that only a few sanctioned entities can be traced to known services, while a 2-step analysis provides enhanced context to their operations. However, although expanding the analysis to include more steps seems beneficial, it should be noted that this approach also introduces more unlabelled addresses, increasing the uncertainty in the "follow-the-money" investigation. Therefore, the number of steps to be included must be determined on a case-by-case basis.

Limitation. When interpreting the results, it is important to take into account some assumptions/constraints considered during this investigation. Firstly, the results of the *behavioural analysis* strongly depend on the quantity and quality of labelled data available in the literature. The outcomes of this work are strictly related to the information provided by the US OFAC authority and need to be verified when information from new authorities becomes available. At the same time, regarding the quantity of the data, as introduced in Section 3.2, this work has gathered 38M addresses of almost 400 entities used in many state-of-the-art works [24][28][32]. Yet, these 38M represent only 2.9% of the addresses used in the BTC blockchain, which counts about 1,300M addresses as of February 2024. Additionally, the OFAC SDN list being considered includes 40 sanctioned entities with Bitcoin addresses involved. While one might argue that this number is insufficient for comprehensive trend analysis, it should be noted that this is the maximum number available in the real ecosystem. On the other hand, regarding the quality of the information, we rely on the fact that the labelled datasets are used in many research investigations, as mentioned in section 3.2, and in some cases, they are extracted by tools that are used by LEAs in real investigations [18]. Moreover, although these datasets are generated and gathered from different sources, they do not present inconsistencies among them in the provided data.

Furthermore, it is to be noted that some entities were sanctioned in late 2023 or early 2024, meaning that the gathered crypto transactions (until February 2024) might not fully capture their activities. However, we decided to focus the analysis on the type of violation rather than concrete entities (Table 2). In fact,

looking at the distribution analysis provided in Figure 2c, it is possible to see that the majority of the entities for each violation are prior to May 2023 - excluding the *ILLICIT-DRUGS* crime - allowing us to analyse 9 months of transactions.

Finally, in the context of the 2-step analysis, this study has assumed that there has been no change in ownership of the funds. However, relying just on blockchain information, the validity of this assumption cannot be ensured. Nonetheless, we acknowledge this risk because limiting the analysis solely to 1-step graphs would be excessively restrictive, generating a skewed and poor view of the impact of the sanctions. Furthermore, a change in the ownership of the funds should not generate high-impact deviations in the proposed analysis, since we are just analysing the reached entities using a basic "follow-the-money" approach.

5 Conclusion

The present study represents a first step and provides an interesting yet partial understanding of the impact of sanctions on the crypto ecosystem, with a focus on the Bitcoin cryptocurrency. The first point to emphasise is that sanctions are inherently tied to the prevailing regulations at any given time. As context, this study solely reports on European policies aimed at regulating the crypto market and preventing financial fraud. However, the analysis is conducted using data provided by the US OFAC authority, as it is the only entity that releases a comprehensive list of economically and trade-sanctioned entities.

The *flow analysis* shows that in general, sanctions have been effective on at least half of the sanctioned entities, while the other half has continued to move (receive and send) money through the sanctioned BTC addresses, although they do not show huge changes in their current balance. In particular, sanctions seem to be not very effective against entities related to specific violations, like *CYBER2* and *ILLICIT-DRUGS* which are the ones that still make transactions and move high quantities of USD in proportion with their activities pre-sanctions. On the other hand, the *behavioural analysis* highlights that sanctioned entities tend to prefer reaching out directly to *Exchanges* to convert their cryptos, rather than use dedicated services for money laundering (mixers or gambling).

With the aim of deepening that understanding, further analysis should include heuristics assumptions in the loop [31] as well as automatic labelling strategies to generate clustered entities. Yet, in this way, it would be possible for each punished entity, not only to consider the actual sanctioned addresses reported in the list, but also other addresses that are likely to belong to the same wallet. At the same time, it will be interesting to scale up our approach by incorporating information from other cryptocurrencies (Ethereum) and stablecoins. In fact, as reported in [12], they represent a good alternative - especially in the last three years - through which criminals engage in illicit activities and sanctions circumvention. This approach will allow Law Enforcement Officers to have a complete picture of criminal *modus operandi*.

Acknowledgments. This work has been partially supported by the European Union's Horizon 2020 Research and Innovation Program under the project FALCON (Grant Agreement No. 101121281)

References

1. Directive (eu) 2015/849 of the european parliament and of the council on the prevention of the use of the financial system for the purposes of money laundering or terrorist financing (2015), https://eur-lex.europa.eu/legal-content/EN/TXT/uri=celex\%3A32015L0849, accessed on 08/05/2024
2. Directive of the european parliament and of the council on the fight against fraud to the union's financial interests by means of criminal law (2017), https://www.consilium.europa.eu/en/policies/fight-against-terrorism/fight-against-terrorist-financing/, accessed on 08/05/2024
3. Directive of the european parliament and of the council on combating fraud and counterfeiting of non-cash means of payment (2019/713), https://eur-lex.europa.eu/eli/dir/2019/713/oj, accessed on 08/05/2024
4. International standards on combatting money laundering and the financing of terrorism and proliferation, the fatf recommendations (2023), https://www.fatf-gafi.org/content/dam/fatf-gafi/recommendations/FATF\%20Recommendations\%202012.pdf.coredownload.inline.pdf, accessed on 08/05/2024
5. Regulation of the european parliament and of the council on markets in crypto-assets (2023), https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX\%3A32023R1114&pk_campaign=todays_OJ&pk_source=EURLEX&pk_medium=TW&pk_keyword=Crypto\%20assets&pk_content=Regulation&pk_cid=EURLEX_todaysOJ, accessed on 08/05/2024
6. Regulation of the european parliament and of the council on information accompanying transfers of funds and certain crypto-assets (2023/1113), https://eur-lex.europa.eu/eli/reg/2023/1113/oj, accessed on 08/05/2024
7. Cryptocurrency bans explained: Which countries have restricted crypto and why? (2024), https://www.techopedia.com/cryptocurrency-bans-explained-which-countries-have-restricted-crypto, accessed on 25/05/2024
8. Alotibi, J., Almutanni, B., Alsubait, T., Alhakami, H., Baz, A.: Money laundering detection using machine learning and deep learning. International Journal of Advanced Computer Science and Applications **13**(10) (2022)
9. Bele, J.L.: Cryptocurrencies as facilitators of cybercrime. In: SHS Web of Conferences. vol. 111, p. 01005. EDP Sciences (2021)
10. Berry, H.S.: The evolution of cryptocurrency and cyber attacks. In: 2022 International Conference on Computer and Applications (ICCA). pp. 1–7 (2022). https://doi.org/10.1109/ICCA56443.2022.10039632
11. Blanchini, M., Cerreta, M., Di Monda, D., Fabbri, M., Raciti, M., Ahmad, H.S., Costa, G.: Supporting criminal investigations on the blockchain: A temporal logic-based approach (2022)
12. Chainalysis Inc.: The 2024 Crypto Crime Report (2024), \url{https://go.chainalysis.com/crypto-crime-2024.html}, accessed on 07/08/2024
13. Cole, T., Gundur, R.: Virtual currency, cryptoassets, and cybercrime. In: Oxford Research Encyclopedia of Criminology and Criminal Justice (2024)
14. Connolly, L.Y., Wall, D.S.: The rise of crypto-ransomware in a changing cybercrime landscape: Taxonomising countermeasures. Computers & Security **87**, 101568 (2019)

15. Fleder, M., Kester, M.S., Pillai, S.: Bitcoin transaction graph analysis. arXiv preprint arXiv:1502.01657 (2015)
16. Guidara, A.: Cryptocurrency and money laundering: A literature review. Corporate Law & Governance Review **4**(2), 36–41 (2022)
17. Hornuf, L., Momtaz, P.P., Nam, R.J., Yuan, Y.: Cybercrime on the ethereum blockchain (2023)
18. Iknaio: Blockchain Analytics Pilot in Bavaria (2022), `\url{https://www.ikna.io/2022/06/22/ZCB.html}`, Accessed on 23/05/2024
19. Japinye, A.: Integrating machine learning in anti-money laundering through crypto: A comprehensive performance review. European Journal of Accounting, Auditing and Finance Research **12**(4), 54–80 (2024)
20. Lin, D., Wu, J., Yu, Y., Fu, Q., Zheng, Z., Yang, C.: Denseflow: Spotting cryptocurrency money laundering in ethereum transaction graphs. In: Proceedings of the ACM on Web Conference 2024. p. 4429–4438. WWW '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3589334.3645692, `https://doi.org/10.1145/3589334.3645692`
21. Nazzari, M.: From payday to payoff: Exploring the money laundering strategies of cybercriminals. Trends in Organized Crime pp. 1–18 (2023)
22. Nazzari, M.: Lost in the maze: Disentangling the behavioral variety of money laundering. European Journal on Criminal Policy and Research pp. 1–19 (2023)
23. Neumann, M., Elsenbroich, C.: Introduction: the societal dimensions of organized crime. Trends in Organized Crime **20**, 1–15 (2017)
24. Paquet-Clouston, M., Haslhofer, B., Dupont, B.: Ransomware payments in the bitcoin ecosystem. Journal of Cybersecurity **5**(1), tyz003 (2019)
25. Reddy, E., Minnaar, A.: Cryptocurrency: A tool and target for cybercrime (12 2018)
26. Sanusi, K.A., Dickason-Koekemoer, Z.: Cryptocurrency returns, cybercrime and stock market volatility: Gas and regime switching approaches. International Journal of Economics and Financial Issues **12**(6), 52 (2022)
27. Soudijn, M.: Encounters with professional money launderers; an analysis of financial transactions as reported by gatekeepers. European Journal on Criminal Policy and Research pp. 1–14 (2024)
28. Tovanich, N., Cazabet, R.: Fingerprinting bitcoin entities using money flow representation learning. Applied Network Science **8**(1), 63 (2023)
29. Viscusi, W.K.: Market incentives for criminal behavior. In: The Black youth employment crisis, pp. 301–351. University of Chicago Press (1986)
30. Wardani, A., Ali, M., Barkhuizen, J.: Money laundering through cryptocurrency and its arrangements in money laundering act. Lex Publica **9**(2), 49–66 (2022)
31. Zhang, Y., Wang, J., Luo, J.: Heuristic-based address clustering in bitcoin. IEEE Access **8**, 210582–210591 (2020)
32. Zola, F., Eguimendia, M., Bruse, J.L., Urrutia, R.O.: Cascading machine learning to attack bitcoin anonymity. In: 2019 IEEE International Conference on Blockchain (Blockchain). pp. 10–17. IEEE (2019)

CBT Session 3: Cryptography for cryptocurrencies

Practical Implementation of Pairing-Based zkSNARK in Bitcoin Script

Federico Barbacovi[1], Enrique Larraia[1], Paul Germouty*, and Wei Zhang[1]

nChain
f.barbacovi@nchain.com, e.larraia@nchain.com, germouty.paul@orange.fr, w.zhang@nchain.com

Abstract. Groth16 is a pairing-based zero-knowledge proof scheme that has a constant proof size and an efficient verification algorithm. Bitcoin Script is a stack-based low-level programming language that is used to lock and unlock bitcoins. In this paper, we present a practical implementation of the Groth16 verifier in Bitcoin Script deployable on the mainnet of a Bitcoin blockchain called BSV. Our result paves the way for a framework of verifiable computation on Bitcoin: a Groth16 proof is generated for the correctness of an off-chain computation and is verified in Bitcoin Script on-chain. This approach not only offers privacy but also scalability. Moreover, this approach enables smart contract capability on Bitcoin which was previously thought rather limited if not non-existent.

Keywords: Bitcoin · Smart Contract · Zero-Knowledge Proof.

1 Introduction

Zero-knowledge proofs (ZKPs) have been widely adopted to enhance blockchain technology. For example, zCash [36] and Firo [16] use ZKPs for user privacy, and Ethereum uses ZKPs for scalability [14]. Bitcoin, on the other hand, is thought to have limitations that make ZKP integration much more difficult, one of which is the Bitcoin scripting language. Due to its stack-based structure and primitive set of opcodes, it is rather difficult to implement the complex mathematical functions required by most ZKPs.

Despite these limitations, there are many projects trying to integrate ZKPs and Bitcoin. They determined to make Bitcoin more scalable and more capable for smart contracts, ultimately making Bitcoin more economically sustainable and viable even with the diminishing block reward through halving. For examples, B^2 Network [4] and Merlin Chain [23] are working on ZK Rollups to scale Bitcoin; Bitlayer [9] takes a BitVM [26] approach to create a computational layer for Bitcoin, while LumiBit [21] adapts ZKEVM to achieve the same. In [27], ZeroSync has compressed the bootstrapping process (initial block download) into a single ZKP verification using a ZKP-circuit-friendly language called Cairo [13], thus dramatically reducing the time required to start a Bitcoin node. However,

* Formerly nChain researcher. Work done while at nChain.

in all the examples mentioned above, practically verifying ZKP on-chain has not yet been realised.

sCrypt [31] has implemented ZKP verification (Groth16, [19]) in Bitcoin Script on the BSV, a version of Bitcoin, achieving a script size of 1.2MB [32,33]. However, their implementation is not practical, and it falls short for three reasons. First, it cannot be deployed on the BSV mainnet without a collaborating miner. This is because of a policy that restricts the script size to 500kB [25], and non-policy-compliant transactions can only be accepted by other miners if the collaborating miner successfully mines a block. Second, it is not a faithful implementation of the Groth16 verifier as they hard-code in the script data which should be revealed at the point of spending, thus greatly limiting the applicability of their code. Third, their approach of using a compiler that converts TypeScript to Bitcoin Script generally leads to scripts of non-optimal size.

Our main contributions are:

- an implementation of bilinear pairings in Bitcoin Script, which has size[1] of 293.6kB, and that can be readily used on the BSV mainnet;
- an implementation of Groth16 verification in Bitcoin Script, which has size[2] of 466kB, and that can be readily used on the BSV mainnet;
- an analysis of the trade-off between script size and execution time caused by large number arithmetics (the smaller the script, the larger the numbers, hence the longer execution time);
- a significant reduction in transaction fees for on-chain ZKP verification as the transaction size is significantly reduced.

To achieve this level of optimisation, we use a combination of different techniques, each providing a significant reduction in script size:[3]

- stack management: being aware of positions of elements on the stacks and identifying the best arrangement of data elements and operations that results in the smallest script;
- no computation of inverses: designing the script to verify a candidate inverse instead of computing it;
- sparseness: working with field elements represented by polynomials that have many zero coefficients [30];
- seed choice: choosing an elliptic curve with seed having the smallest Hamming weight.

[1] All the sizes are cumulative of the locking and unlocking script size.

[2] The size reported here is for one public input, in Section 4 we also report the size for the Groth16 verifier with two public inputs.

[3] It is difficult to pinpoint where exactly the reductions come from, as they are a combination of all the techniques we employed. However, in the body of the text we provide rough estimations for each of the techniques we employ.

Our scripts are publicly available on GitHub.[4]. In the repository, readers can find a Python code used to generate our scripts,[5] another Python code used to generate the test data for benchmarking purposes, and the references to examples of on-chain transactions.

Our results enable, for the first time, practical ZKP verification on the mainnet of a Bitcoin version called BSV. [6] While Ethereum uses ZKPs for scalability, Bitcoin can also use them to enable smart contracts with greater flexibility. That is, in theory, one can run any computation off-chain and generate a ZKP, which is verified on chain, that the computation was done correctly. For example, the computation can be the validation of a token rule set, the enforcement of a financial contract, or the execution of instructions based on business logic and workflows.

The paper is structured as follows. In Section 2, we recall the preliminary notions we need in the rest of the paper, and we introduce the notation we use for Bitcoin scripts. In Section 3, we detail our implementation of the Optimal Ate Pairing and of the Groth16 verifier instatiated over the curve BLS12-381. Finally, in Section 4 we evaluate our scripts according to script size and execution time, and we compare the cost of verifying a ZKP on BSV and on Ethereum.

2 Preliminaries

2.1 Bitcoin

The Bitcoin blockchain [28] parses block data x into an ordered set of transactions $x := (\mathsf{tx}_1, \ldots, \mathsf{tx}_n)$. Each transaction specifies a list of inputs and outputs. An output of a transaction is *spent* if it is referenced as an input of a valid transaction. An output can only be spent once.

Bitcoin uses a non-Turing complete, stack-based programming language in which spending conditions can be coded into *locking* scripts contained in outputs. Each input of a transaction contains an *unlocking* script, with the arguments needed to execute the locking script from the output referenced by the input. The spending is accepted if the execution terminates with true.

We think of the subroutines that make up a locking script as the implementations of functions $f(x_1, \ldots, x_n)$, and of the elements in the unlocking script as the values $(\tilde{x}_1, \ldots, \tilde{x}_n)$ over which the functions are evaluated. The unlocking script is then the implementation of a predicate (a function that returns true or false) whose calculation requires the evaluation of various functions (the subroutines) on the values supplied in the unlocking script.

[4] https://github.com/nchain-innovation/zkscript_package

[5] We generate our scripts as outputs of Python functions, so that they can be composed and shuffled around in an easy way. The approach we take is similar to that of BitVM [12], but they use Rust in place of Python.

[6] For our optimisations to work on BTC, we need large integer arithmetic.

Opcode	Operation
OP_1SUB	$x_0 - 1 \leftarrow x_0$ OP_1SUB
OP_DEPTH	$x_n, \ldots, x_0, n+1 \leftarrow x_n, \ldots, x_0$ OP_DEPTH
OP_PICK	$x_n, \ldots, x_0, x_i \leftarrow x_n, \ldots, x_0, i$ OP_PICK
OP_EQUAL	$x_0 == x_1 \leftarrow x_0, x_1$ OP_EQUAL
OP_VERIFY	Pop x_0; fail if x_0 is false, otherwise, continue $\leftarrow x_0$ OP_VERIFY
OP_EQUALVERIFY	OP_EQUALVERIFY = OP_EQUAL OP_VERIFY

Table 1. Opcodes used in this paper

Script execution A script is a sequence of opcodes and data objects. It is evaluated in reverse Polish notation by the Bitcoin Script engine, starting by pushing to the stack the arguments specified in the unlocking script.

The set of opcodes available for scripting depends on the implementation of Bitcoin. We will use the BSV implementation [11] because it supports large numbers (of size up to 10kB) and has the widest opcode support for arithmetic operations.[7] We list the opcodes used in this paper in Table 1; note that they are not all the ones needed in our implementations.

We introduce some notation that will be used throughout this work:

- $\langle\langle l \rangle\rangle$ denotes data hard-coded in the locking script. Thus, we will write

$$[\texttt{foo}] \coloneqq \langle\langle l \rangle\rangle [\texttt{bar}]$$

  to denote that the locking script [foo] consists of a subroutine script [bar] and hard-coded data $\langle\langle l \rangle\rangle$.
- x denotes data on top of the stack, i.e., the data we would get if we popped an element from the stack. More generally, $x_0, \ldots, x_n$ means that x_n is the data on top of the stack, x_{n-1} is the data below x_n (second from the top), and x_0 is the data buried at depth $n+1$ in the stack.
- $y_0, \ldots, y_m \leftarrow x_0, \ldots, x_n$ [foo]: Before executing [foo], the top $n+1$ elements of the stack are $x_0, \ldots, x_n$, and after executing [foo] the top $m+1$ elements are $y_0, \ldots, y_m$.

2.2 Pairings

Bilinear pairings are the building block of many important cryptographic primitives. The most efficient instantiation of a bilinear pairing is the Optimal Ate pairing [22], which is defined as a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that

$$e(n[A], [B]) = e([A], n[B]) = e([A], [B])^n$$

[7] BTC only allows number up to 4 bytes and has disabled various opcodes needed for arithmetic operations [1].

for any $[A] \in \mathbb{G}_1$, $[B] \in \mathbb{G}_2$ and $n \in \mathbb{Z}$. Here, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of the group of points on some elliptic curves.

The value of e on $(P, Q) \in \mathbb{G}_1 \times \mathbb{G}_2$ is computed in two steps. First, we compute $\mathsf{miller}(P, Q) \in \mathbb{G}_T$, the output of the Miller loop [24], and then we perform a final exponentiation $\mathsf{miller}(P, Q)^\eta$, where η is an exponent depending on the instantiation of $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$. We set $e(P, Q) \coloneqq \mathsf{miller}(P, Q)^\eta$.

In this paper, we instantiate the Optimal Ate Pairing over BLS12 curves [8] for their robustness against some known attacks [5–7, 20] and their efficiency [3].

BLS12 curves BLS12 curves have the form $y^2 = x^3 + b \mod q$, where b is a parameter of the curve, and $q = (u-1)^2(u^4 - u^2 + 1)\ 3 + u$ is a prime dependent on a seed u. We write $E_{b,u}(\mathbb{F}_{q^n})$ to denote set of points of the BLS12 curve with parameters b and u over $\mathbb{F}_{q^n}$.

For BLS12-381, we have $u = -(2^{63} + 2^{62} + 2^{60} + 2^{57} + 2^{48} + 2^{16})$ and $b = 4$. When instantiating the Optimal Ate Pairing over this curve, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order $r = u^4 - u^2 + 1$, $\mathbb{G}_1$ is a subgroup of $E_{b,u}(\mathbb{F}_q)$, while $\mathbb{G}_T = \mathbb{F}_{q^{12}}$. To construct $\mathbb{G}_2$, we set $\mathbb{F}_{q^2} = \mathbb{F}_q[t]\ (1+t^2)$, and then $\mathbb{G}_2$ is a subgroup of $E_{b',u}(\mathbb{F}_{q^2})$, where $b' = (1 + t)b \in \mathbb{F}_{q^2}$.

2.3 zkSNARKs

Circuits and NP relations. Let $\mathsf{C} : \mathbb{F}_r^{\ell+h} \to \{0, 1\}$ be a polynomial-size arithmetic circuit over a finite field $\mathbb{F}_r$. The NP relation $\mathcal{R}_\mathsf{C}$ for C is defined as

$$\mathcal{R}_\mathsf{C} \coloneqq \left\{ (\boldsymbol{a}; \boldsymbol{w}) \in \mathbb{F}_r^\ell \times \mathbb{F}_r^h \mid \mathsf{C}(\boldsymbol{a}, \boldsymbol{w}) = 1 \right\}.$$

The vector $\boldsymbol{a} = (a_1, \ldots, a_\ell)$ is the statement of the relation, sometimes also called the instance or public input, and the vector $\boldsymbol{w}$ is the witness or private input. The language associated to $\mathcal{R}_\mathsf{C}$ is $\mathcal{L}_\mathsf{C} \coloneqq \{\boldsymbol{a} \in \mathbb{F}_r^n \mid \exists \boldsymbol{w} \in \{0, 1\}^h \text{ s.t. } (\boldsymbol{a}; \boldsymbol{w}) \in \mathcal{R}_\mathsf{C}\}$.

zkSNARKs A preprocessing, zero-knowledge, succinct, non-interactive, argument system of knowledge (zkSNARK[8]) for $\mathcal{R}_\mathsf{C}$ is a triplet of algorithms $\Pi \coloneqq (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ such that Setup takes as input a security parameter λ and the description of the circuit C, and it outputs a pair of keys pk and vk. The prover Prove takes pk, the statement $\boldsymbol{a}$ and the witness $\boldsymbol{w}$ and outputs a proof $\pi \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_1$, that is $\pi \leftarrow \mathsf{Prove}(pk, \boldsymbol{a}, \boldsymbol{w})$. The verifier Verify takes vk, $\boldsymbol{a}$, π and it either accepts or rejects, that is $\{\mathsf{true}, \mathsf{false}\} \leftarrow \mathsf{Verify}(vk, \boldsymbol{a}, \pi)$. The proof purportedly is for the statement "$\boldsymbol{a} \in \mathcal{L}_\mathsf{C}$".

Groth16 Groth16 [19] follows the linear interactive proof paradigm [10] with security in the generic group model, and can thus be instantiated with pairings. Given $\pi = ([A]_1, [B]_2, [C]_1)$, the result of Verify on the public input

[8] In this paper, we use ZKP and zkSNARK interchangeably. It is understood that there is a subtle difference which is not relevant to this paper.

$\boldsymbol{a} = (a_1, \ldots, a_\ell)$ and the proof π is the result of an equation of pairings:

$$e([A]_1, [B]_2) = e([\alpha]_1, [\beta]_2) \cdot e\left(\sum_{i=0}^{\ell} a_i [P_i]_1, [\gamma]_2\right) \cdot e([C]_1, [\delta]_2) \tag{1}$$

where $[\alpha]_1, [\beta]_2, [\gamma]_2, [\delta]_2$ and the $[P_i]_1$'s[9] are part of vk, and $a_0 = 1$.

Note that the equation (1) depends on C only via the number of public inputs. This means that a Bitcoin Script implementation of Verify will be independent of the complexity of the calculations happening in C, and will scale only according to the number of public inputs. Furthermore, the size of the proof π is fixed to that of three groups element, once again independent of C.

These properties make Groth16 the best candidate for implementing zk-SNARK verification on-chain as it induces the least transaction size and computational complexity, resulting in low fees and fast execution time.

3 Implementation of Pairings and Groth16 in Script

We now outline our implementation of the Optimal Ate Pairing and of the Groth16 verifier in Bitcoin Script. First, we break down the scripts into subroutines. That is, we look at the operations required to compute the Optimal Ate Pairing and to verify a Groth16 proof, and we decompose these operations into simpler ones that we efficiently implement. Second, in constructing the subroutines we assume that the spender (the prover in the zkSNARK framework) supplies to the script (the verifier) additional input data to simplify the proof verification. The script will then verify that the data supplied is the one purported to be, and use it if it is, or fail otherwise. See Section 3.1 for an example of the input data supplied by the spender.

Remark 1. In Bitcoin, the prover/verifier framework of zkSNARKs is transposed to that of a payer who constructs the locking script (the Groth16 verifier) and of a spender who shows a zero-knowledge proof to prove they have the right to spend. While in zkSNARKs only the verifier carries the burden of verifying the proof, in our setup we assume that also the prover performs part of the computation required to verify the proof, so to reduce the size of the Bitcoin Script Groth16 verifier. This means that there is an additional burden on the prover, which is quantified by the amount of work required to compute the product of the three Miller loops in (6). This added burden is acceptable as it is easier to increase efficiency of off-chain computations rather than optimising script size.

The Optimal Ate Pairing is computed as $e(P, Q) = \mathsf{miller}(P, Q)^\eta$, see Section 2.2. We implement it by first implementing the Miller loop, and then the final exponentiation. That is, we construct a script [`millerLoop`] that computes

[9] The $[P_i]$'s are the evaluations of the QAP polynomials of the public inputs corresponding to the R1CS system of C on the verifier pre-computed challenge (the so-called *toxic waste* generated in the setup), [18].

the function $(P, Q) \mapsto \mathsf{miller}(P, Q)$, and a script $[\texttt{finalExponentiation}]$ that computes the function $(-) \mapsto (-)^\eta$. More precisely, on input data

$$\begin{aligned} [\texttt{inputMillerLoop}] &= \ \texttt{aux}_{\texttt{miller}} \ , \ P \ , \ Q \\ [\texttt{inputFinalExponentiation}] &= \ \texttt{aux}_{\texttt{exp}} \ , \ x_0 \end{aligned} \tag{2}$$

where $x_0 \in \mathbb{F}_{q^{12}}$, and $\mathsf{aux}_{\mathsf{miller}}$, $\mathsf{aux}_{\mathsf{exp}}$ is auxiliary data, they compute

$$\begin{aligned} \mathsf{miller}(P, Q) \ &\leftarrow [\texttt{inputMillerLoop}] \ [\texttt{millerLoop}] \\ x_0^\eta \ &\leftarrow [\texttt{inputFinalExponentiation}] \ [\texttt{finalExponentiation}] \end{aligned} \tag{3}$$

Then the script implementing the Optimal Ate Pairing is

$$[\texttt{pairing}] = [\texttt{millerLoop}] \ [\texttt{finalExponentiation}] \tag{4}$$

Indeed, on input $[\texttt{inputPairing}] = \ \texttt{aux}_{\texttt{exp}} \ , \ \texttt{aux}_{\texttt{miller}} \ , \ P \ , \ Q$, we get

$$e(P, Q) \ \leftarrow [\texttt{inputPairing}] \ [\texttt{pairing}]$$

We approach the Groth16 verifier in a similar way. By rearranging (1) from Section 2.3 using the bilinear properties of e and its definition, we see that Groth16 verification entails verifying the following equation

$$\begin{aligned} &\left(\mathsf{miller}([A]_1, [B]_2) \cdot \mathsf{miller}\left(\sum_{i=0}^{\ell} a_i [P_i]_1, -[\gamma]_2\right) \cdot \mathsf{miller}([C]_1, -[\delta]_2)\right)^\eta \\ &= e([\alpha]_1, [\beta]_2) \end{aligned} \tag{5}$$

Hence, we need a script $[\texttt{multiScalarMultiplication}]$ that computes the function $(a_1, \ldots, a_\ell) \mapsto \sum_{i=0}^{\ell} a_i [P_i]_1$, and a script $[\texttt{tripleMillerLoop}]$ that computes the product of the three Miller loops in (5). Then, the Groth16 verifier is[10]

$$\begin{aligned} [\texttt{groth16Verifier}] = \ &[\texttt{multiScalarMultiplication}] \\ &[\texttt{tripleMillerLoop}] \ [\texttt{finalExponentiation}] \\ &\langle\!\langle e([\alpha]_1, [\beta]_2) \rangle\!\rangle \ \texttt{OP\_EQUALVERIFY} \end{aligned} \tag{6}$$

We now detail the challenges to implement the subroutines the make up $[\texttt{pairing}]$ and $[\texttt{groth16Verifier}]$, and our proposed solutions.

3.1 Optimising the Miller loop

The value $\mathsf{miller}(P, Q)$ for curves in the BLS12 family is computed according to algorithm (1), where $\mathrm{ev}_{\ell_{T,Q}}(P)$ denotes the evaluation of the line through T and Q at P.

[10] Note that $e([\alpha]_1, [\beta]_2)$ can be hard-coded because $[\alpha]_1, [\beta]_2$ are part of vk and are known before the proof is generated.

Algorithm 1 Miller Loop

Inputs: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, $u = \sum_{i=0}^{n} u_i 2^i$, $u_i \in \{-1, 0, 1\}$, $u_n \neq 0$
Output: $\mathsf{miller}(P, Q) \in \mathbb{G}_T$

$\mathsf{out} \leftarrow 1$
if $u_n = 1$ **then**
  $T \leftarrow Q$
else
  $T \leftarrow -Q$
end if
for $i = n-1, \ldots, 0$ **do**
  $\mathsf{out} \leftarrow \mathsf{out}^2$
  $T \leftarrow 2T$
  if $u_i = 1$ **then**
    $\mathsf{out} \leftarrow \mathsf{out} \cdot \mathrm{ev}_{\ell_{T,Q}}(P)$
    $T \leftarrow T + Q$
  else
    $\mathsf{out} \leftarrow \mathsf{out} \cdot \mathrm{ev}_{\ell_{T,-Q}}(P)$
    $T \leftarrow T - Q$
  end if
end for

Seed choice To obtain the most efficient implementation of [`millerLoop`], we seek to minimise the length of the loop and the cost of performing the operations in each iteration. The length of the loop and the number of operations performed can be minimised by choosing a curve whose seed u has small Hamming weight (number of non-zero bits) and bit-length. Our choice is BLS12-381, for which u has bit-length 64 and Hamming weight equal to 6, see Section 2.2.

Sparseness. The number of operations performed in the Miller loop can be further reduced by leveraging *sparseness* as explained by Scott in [29]. Both out and the line evaluations belong to the finite field extension $\mathbb{F}_{q^{12}}$, but many of the coefficients of the line evaluations are zero (that is why Scott calls them sparse). Leveraging this knowledge, we reduce the size of the script required to multiply two line evaluations from 1kB (the size of our implementation of multiplication over $\mathbb{F}_{q^{12}}$, to 150 bytes (the size of our script for the multiplication of two sparse elements).

Remark 2. When implementing [`tripleMillerLoop`] for (6), instead of computing $\mathsf{miller}([A]_1, [B]_2)$, $\mathsf{miller}\left(\sum_{i=0}^{\ell} a_i [P_i]_1, -[\gamma]_2\right)$ and $\mathsf{miller}([C]_1, -[\delta]_2)$ one after the other, we parallelise the computation. Namely, as evaluating $\mathsf{miller}(-,-)$ means executing algorithm (1), instead of repeating the loop three times, we go through the loop once, and at every iteration we carry out the computations required by each of the three terms appearing in (6). In this way, we can multiply together the sparse elements coming from the various line evaluations, thus amplifying the size optimisation resulting from leveraging sparseness.

Verifying the gradient. Finally, we look at reducing the cost of performing the various operations required by the Miller loop. To update the value of T, we sum points in $\mathbb{G}_2 \subset E_{b',u}(\mathbb{F}_{q^2})$, while to update $\texttt{out}$, we need to compute line evaluations. The most inefficient part of these operations is the calculation of the gradient of the line through two points on the curve. The inefficiency is due to the fact that computing the gradient requires inverting an element in a finite field, an operation whose cost in Bitcoin Script is substantial.[11]

To avoid the overhead of computing the gradient on-chain, we verify a candidate provided in the unlocking script as part of the auxiliary data $\texttt{aux}_{\texttt{miller}}$. Namely, every time we need the gradient of the line through two points $R_1, R_2 \in \mathbb{G}_2$, we expect the gadient $\lambda \in \mathbb{F}_{q^2}$ to be supplied in $\texttt{aux}_{\texttt{miller}}$, and we verify that λ is computed correctely by verifying

$$\lambda \cdot (x_{R_2} - x_{R_1}) = y_{R_2} - y_{R_1}$$

where $R_i = (x_i, y_i) \in \mathbb{F}_{q^2} \times \mathbb{F}_{q^2}$. Note that verification is very efficient, as once λ is verified, it can be used multiple times. Putting it into numbers, verifying the gradient instead of computing it on chain allows us to save roughly $3 \cdot \log(q) = 3 \cdot 381 \sim 1100$ bytes.

3.2 Optimising the final exponentiation

Final exponentiation is the same for [`pairing`] and [`groth16Verifier`], and it entails raising an element of $\mathbb{F}_{q^{12}}$ to the power η. To minimise the script size of [`finalExponentiation`], we follow the standard approach in the literature and split the final exponentiation in an easy and a hard part

$$[\texttt{finalExponentiation}] = [\texttt{easyExponentiation}]\ [\texttt{hardExponentiation}]$$

In the hard part, we leverage the Frobenius map $\mathbb{F}_{q^n} \to \mathbb{F}_{q^n}, z \to z^q$, to fix the cost of [`hardExponentiation`] to (roughly) that of performing five exponetiations to the power u.

For the easy part, we need to compute one Frobenius map, and to invert an element in $\mathbb{F}_{q^{12}}$. Instead of performing inversion on-chain, similarly to what we did in the Miller loop, we verify an inverse candidate supplied in the unlocking script as part of the auxiliary data $\texttt{aux}_{\texttt{exp}}$. Namely, as we need the inverse of an element $z \in \mathbb{F}_{q^{12}}$, we expect the inverse z' to be supplied in $\texttt{aux}_{\texttt{exp}}$, and on-chain we verify $z \cdot z' = 1 \in \mathbb{F}_{q^{12}}$. This allows us to fix the cost of [`easyExponentiation`] to constant (it is independent of the curve parameters).

Remark 3. Even if the Frobenius map entails raising an element to the power q, its implementation is of constant size because it only requires multiplying the components of $z \in \mathbb{F}_{q^n}$ by some constants.

[11] If $z \in \mathbb{F}_q$, then inverting z in Bitcoin Script requires $O(\log(q))$ operations using Fermat's Little Theorem.

3.3 Optimising the multi scalar multiplication

The hardest subroutine to optimise in (6) is [`multiScalarMultiplication`]. The reason is that this subroutine computes $\sum_{i=0}^{\ell} a_i[P_i]_1$, which depends both on circuit-specific values: the $[P_i]'s$, see Section 2.3, and on values supplied by the prover: the public inputs $a_1, \ldots, a_\ell$.

As the public inputs are supplied by the prover, they are not known when [`multiScalarMultiplication`] is constructed, and therefore the script must take into account the worst case scenario, namely, $a_i = r$.

The cost of computing $\sum_{i=0}^{\ell} a_i[P_i]_1$ scales linearly with ℓ, which is unfortunate as a single multiplication $a_i[P_i]_1$ costs about 35kB via double-and-add (and verifying the gradient as in Section 3.1). To optimise the size of the script we use a standard trick: pass the ℓ public inputs a_i of C as witness and a hash of them as public input. This makes the size of the script independent of the number of public inputs at the cost of increasing the computational burden of the prover.

More specifically, let $H : \{0,1\}^* \to \mathbb{F}_r^d$ be a cryptographic hash function where $\mathbb{F}_r$ is the field over which C is defined. Then, the augmented relation for which we prove satisfiability is

$$\mathcal{R}' := \left\{ ((h_1, \ldots, h_d); (a_1, \ldots, a_\ell, \boldsymbol{w})) \;\middle|\; \begin{array}{l} \mathsf{C}(a_1, \ldots, a_\ell, \boldsymbol{w}) = 1 \\ (h_1, \ldots, h_d) = H(a_1, \ldots, a_\ell) \end{array} \right\}.$$

In this way, we keep the size of the script fixed to that of a Groth16 verifier for a circuit with d public inputs. Indeed, in a proof for relation $\mathcal{R}'$ the prover supplies d public inputs $h_1, \ldots, h_d$ for which the circuit corresponding to $\mathcal{R}'$ verifies that $(h_1, \ldots, h_d)$ is the digest of $(a_1, \ldots, a_\ell)$, the original public inputs that are now passed as private inputs, and that $\mathsf{C}(a_1, \ldots, a_\ell, \boldsymbol{w}) = 1$. The public inputs $h_1, \ldots, h_d$ are a commitment to the public inputs $a_1, \ldots, a_\ell$.

For example, if $\ell > 2$ we can set H to be the (vector) Pedersen hash over the JubJub curve [34], whose base field is the scalar field $\mathbb{F}_r$ of BLS12-381. A Pedersen hash digest is just a single group element of JubJub $h = (h_1, h_2) \in \mathbb{F}_r^2$. Thus, $d = 2$ and the verification script only needs to compute $h_1[P_1'] + h_1[P_2']$, instead of an ℓ-multi scalar multiplication.

3.4 Subroutine-independent optimisations

In this section we detail some optimisations that we apply to all the subroutines appearing in (4) and (6).

Stack management Stacks are data structures equipped only with push and pop operations, which means that we can only access the top element of the stack. This property makes storage and retrieval of temporary variables a task with great impact on script size.

During script execution, the Bitcoin Script Engine has two stacks at its disposal, the main stack, also referred to as the stack, and the altstack. One can

only push and pull elements from the altstack, which is why it is customary to use it to store variables. We take a different approach, we use the bottom of the stack instead. As the depth of the stack can be obtained with the opcode `OP_DEPTH`, the bottom of the stack can be thought to have a fixed position, and can be used to store variables.

The variable we need more often in [`pairing`] and [`groth16Verifier`] is q, which we store at the bottom of the stack, and fetch with the following script

$$[\texttt{fetch}_q] = \texttt{OP\_DEPTH OP\_1SUB OP\_PICK} \tag{7}$$

In this way, we save ~ 50 bytes compared to pushing q to the stack every time we need it.

Remark 4 (Make fetching secure). As there is no way to efficiently push an element to the bottom of the stack, we assume q is supplied in the unlocking script as part of the auxiliary data. To ensure that it is the one we assume it to be, i.e., the parameter q of BLS12-381, we use the following script

$$[\texttt{verify}_q] = \texttt{OP\_DEPTH OP\_1SUB OP\_PICK} \langle\langle q \rangle\rangle \texttt{ OP\_EQUALVERIFY}$$

Arithmetic over finite fields All the subroutines in (4) and (6) require arithmetic over (a finite field extension of) $\mathbb{F}_q$. The biggest impact of finite field arithmetic on script size comes from modulo operations by q. To efficiently mod by q, we employ two techniques.

First, as taking the residue class modulo q is a homomorphism $\mathbb{Z} \to \mathbb{F}_q$, instead of taking a modulo after every operation, we do it only once in a while. A similar approach was taken in [17], but we improve it by using the *modulo threshold*, i.e., the upper bound on the size of the numbers during script execution, as a parameter of the script. Tuning this parameter we have a trade-off between script size and execution time, see Section 4.

Second, we batch modulo operations, so that q must be fetched only once. We explain the technique in the case of addition, but it can be applied to any other operation. As elements of $\mathbb{F}_{q^n}$ are given by tuples $(z_1, \dots, z_n)$ of elements in $\mathbb{F}_q$, computing $(z_1, \dots, z_n) + (\tilde{z}_1, \dots, \tilde{z}_n)$ means computing $z_i + \tilde{z}_i \mod q$ for $i = 1, \dots, n$. Being Bitcoin Script a stack-based language, we must compute each component $z_i + \tilde{z}_i \mod q$ and place it on top of the stack. Instead of sequentially computing $z_i + \tilde{z}_i \mod q$ for $i = 1, \dots, n$, we compute $z_i + \tilde{z}_i$ for $i = n, \dots, 1$, place them on the altstack, and then sequentially take the modulo of each element. With this technique, we save $(n-1)$ bytes for every modulo operation.

Remark 5 (Preventing overflows). As we remarked in Section 2, the BSV implementation supports large numbers. However, policy restrictions dictate that the numbers must fit in 10kB. To avoid overflows, we proceed as follows. As the operations executed in [`groth16Verifier`] are fixed, and we know the largest size of the input data fed to the script, when constructing the script we keep track of the size of the numbers we are working with. For example, if we multiply

two numbers of bit size at most $|q|$, we know that the result has bit size at most $2|q|$. Then, in the script we reduce modulo q before the numbers overflow.

We go one step further: we introduce a modulo threshold variable that is supplied at the point of script construction and that dictates when to perform modulo operations. See Section 4.1 for more information.

4 Script benchmarking

We now benchmark our scripts according to three metrics: script size, script execution time, and the cost of publishing a transaction with the script on-chain. Based on the first two metrics, we select the optimal modulo threshold, see Section 4.1, and then we compare the monetary cost of executing our script to that of executing an equivalent script on Ethereum, see Section 4.2.

4.1 Script size and execution time

When constructing the [`pairing`] and [`groth16Verifier`], we can choose the threshold after which modulo operations are carried out. Namely, we can choose the largest size the numbers can reach during script execution before we mod by q and bring them back to $\mathbb{F}_q$. Changing the threshold for modulo operations allows us to strike a balance between script size and execution time. Indeed, the more often we mod by q, the bigger the script size, but the lower the execution time of the script, as it will work with smaller numbers.

Below, we plot the threshold for the modulo operations against script size and execution time, respectively, for [`pairing`] and [`groth16Verifier`] with one and two public parameters, i.e., $\ell = 1$ and $\ell = 2$. We run our tests in a BSV regtest v1.0.8 on a processor Intel Core i7, 2.6 Ghz, 6-Core.

While BSV can support transactions with arbitrary script size and execution time, and with numbers of length up to 750kB, current policy restrictions impose that the locking script is at most 500kB, that it executes in at most 1 second, and that the numbers[12] must fit in 10kB [25].

Figure 1 shows that the size of [`pairing`] decreases rapidly when the modulo threshold increases from 50 bytes (which implies that we mod by q after every operation) to 2kB. Further increases of the modulo threshold result in small further decreases of the script size, but at the cost of a higher execution time. In particular, when the modulo threshold reaches 4kB, the execution time approaches the policy threshold of 1 second. As there is not a big difference in either script size or execution time when the modulo threshold passes from 2kB to 3kB, we take a conservative stance and choose 2kB as the optimal modulo threshold for the [`pairing`]. This choice results in a script of size 286kB and with execution time of circa 0.47s.

Figure 2 shows that the size of [`groth16Verifier`] behaves similarly. Namely, the size of the script decreases rapidly when the modulo threshold increases from

[12] By numbers we mean elements on the stack that are used in mathematical operations.

Fig. 1. Size and execution time of [`pairing`] as functions of the modulo threshold

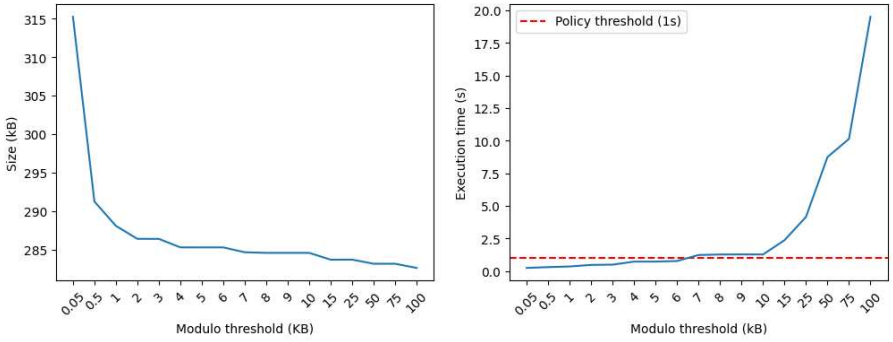
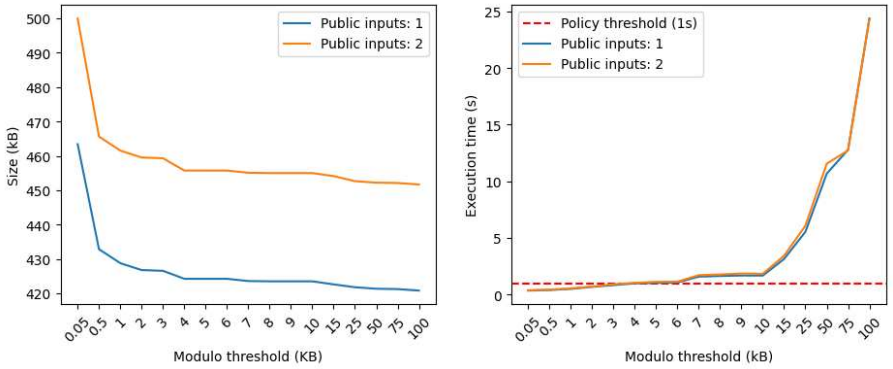


Fig. 2. Size and execution time of [`groth16Verifier`] as functions of the modulo threshold



50 bytes to 2kB, but further increases do not decrease the script size too much. For Groth16, we approach the consensus threshold of 1s execution time when the modulo threshold approaches 3kB. We choose as optimal modulo threshold 2kB, which results in script of sizes 426kB and 460kB, for one and two public inputs, respectively, and with execution times of circa 0.67s and 0.70s, respectively.

Remark 6. In the Figures 1 and 2, we focus on the size of the locking script because the modulo threshold does not affect the size of the unlocking script. However, in Section 4.2 we will take into account both locking and unlocking script, as the cost of publishing of script on-chain depends on both.

4.2 Monetary cost

In this section, we compare the transaction fees for the Optimal Ate Pairing and the Groth16 verifier on BSV and Ethereum, respectively. For simplicity, the comparison only focuses on the cost for the computation to be done by the

nodes in a network. We do not take other factors such as the cost to maintain the respective network or the time it takes for the transaction to be confirmed and published.

The results presented in Table 2 and Table 3 make it apparent that as of May 2024, it is much cheaper to execute bilinear pairings and the Groth16 verifier on BSV than on Ethereum.[13]

In BSV, a miner executes a script [S] = [unlock][lock] if there is a transaction $\mathsf{tx}_{\mathsf{lock}}$ with an output $\mathsf{txOut}_{\mathsf{lock}}$ that has [lock] as locking script, and there is a transaction $\mathsf{tx}_{\mathsf{unlock}}$ with an input $\mathsf{txIn}_{\mathsf{unlock}}$ that spends $\mathsf{txOut}_{\mathsf{lock}}$ and that has [unlock] as unlocking script. In this situation, consensus requires a miner to execute [S], regardless of what operations are contained in it.[14] Thus, we model the cost of executing a script in BSV as the dollar value of the transaction fees required to publish [S] on-chain.

From the analysis of Section 4.1, we see that the optimal modulo thresholds for the Optimal Ate Pairing and the Groth16 verifier are given by 2kB in both cases. The script size for [pairing] and [groth16Verifier] can be read off from Figure 1 and Figure 2, and are 286kB for [pairing], and 426kB, 460kB for [groth16Verifier] with $\ell = 1$, $\ell = 2$, respectively; the size of [unlockPairing], see (2), is 7.6kB, while the size of the unlocking script of [groth16Verifier] with $\ell = 1$ is 40kB, and with $\ell = 2$ is 60kB.

To estimate fee rates on the BSV blockchain and the BSVUSD conversion rate, we download data from WhatsOnChain.com [35]. We consider the average fee rate and the exchange rate for the period going from 15/03/2024 to 12/06/2024. We trim the series by removing the values below the 5th percentile and above the 95th percentile. Then, we take the 25th, 50th, and 75th percentile from the time series of fee rates as estimates of low, medium and high fee rates. They come out to be: 57, 79 and 120 sats/kB, respectively. As estimate of the exchange rate, we take the average price of the trimmed time series, which is \$73 per BSV. The results of the calculations are presented in Table 2 for the Optimal Ate Pairing, and in Table 3 for Groth16 verifier.

The cost to execute an Ethereum contract is proportional to the computational complexity of the underlying code, with computational units measured in terms of gas. Hence, the cost of executing a contract in Ethereum is given by how many units of gas it requires, and gas cost at the time of execution.

Since EIP-1108 [2], the cost of executing one pairing is of 79000 (= 34000 + 45000) gas units, whereas the cost of executing the Groth16 verifier is of 153150 (= $34000 \cdot 3 + 45000 + 6150$, see [2]) gas units for one public statement, and 159300 for two public statements. We estimate the cost of executing the scripts with the same method as for BSV fee rates. We download the data from Etherscan.io [15], and we calculate three fee rates: low: 11 gwei/gas,[15] medium: 16 gwei/gas, and

[13] Note that Ethereum uses a different curve. However, their implementation of pairings is state-of-the-art, so the comparison made here can be considered fair.

[14] The only exception is if the script does not abide by the policies set forth by the miner. For the purpose of this analysis we assume that [S] satisfies such policies.

[15] 1 gwei equates to 10^{-9} ETH.

Blockchain \ Fee rate	Low	Medium	High
BSV	\$0.012	\$0.016	\$0.025
Ethereum	\$2.95	\$4.29	\$6.43

Table 2. Cost of executing the Optimal Ate Pairing in BSV and Ethereum.

	Groth16 verifier ($\ell = 1$)			Groth16 verifier ($\ell = 2$)		
Blockchain \ Fee rate	Low	Medium	High	Low	Medium	High
BSV	\$0.018	\$0.024	\$0.037	\$0.0019	\$0.026	\$0.040
Ethereum	\$5.78	\$8.32	\$12.48	\$5.95	\$8.65	\$12.97

Table 3. Cost of executing the Groth16 verifier in BSV and Ethereum.

high: 24 gwei/gas, as well as an estimated exchange rate of \$3394 per ETH.[16] The results of the calculations are presented in Table 2 for the Optimal Ate Pairing, and in Table 3 for the Groth16 verifier.

5 Conclusion

We have demonstrated not only that it is practical to implement the Groth16 verifier in Bitcoin Script, but also that the cost of executing it is much cheaper than that of executing an equivalent script in Ethereum, see Section 4. As part of our future work, we plan to implement more pairing-based cryptographic primitives in Bitcoin Script so that the Bitcoin blockchain can leverage this fruitful area of cryptography to its full strength.

References

1. Bitcoin (BTC) Wiki, Script. `https://en.bitcoin.it/wiki/Script`
2. Antonio Salazar Cardozo, Zachary Williamson: EIP-1108: Reduce alt_bn128 precompile gas costs," Ethereum Improvement Proposals, no. 1108, May 2018. `https://eips.ethereum.org/EIPS/eip-1108`
3. Aranha, D.F., El Housni, Y., Guillevic, A.: A survey of elliptic curves for proof systems. Designs, Codes and Cryptography **91**(11), 3333–3378 (2023)
4. B Squared: Zero-Knowledge Proof Verification Commitment for ZK-Rollup on Bitcoin. `https://docs.bsquared.network/zpvc`
5. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Journal of cryptology **32**, 1298–1336 (2019)

[16] Note that, even if the exchange rate of BSV per ETH were 1:1, then executing the scripts on BSV would still be much cheaper. For example, at the price of \$3394 per BSV, the Optimal Ate Pairing at high fee rate would cost only \$1.16.

6. Barbulescu, R., Gaudry, P., Guillevic, A., Morain, F.: Improving nfs for the discrete logarithm problem in non-prime finite fields. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 129–155. Springer (2015)
7. Barbulescu, R., Gaudry, P., Joux, A., Thomé, E.: A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–16. Springer (2014)
8. Barreto, P.S., Kim, H.Y., Lynn, B., Scott, M.: Efficient algorithms for pairing-based cryptosystems. In: Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22. pp. 354–369. Springer (2002)
9. Bit Layer: Bitlayer: A Bitcoin Computational Layer Architecture Based on the BitVM Paradigm. `https://static.bitlayer.org/Bitlayer-Technical-Whitepaper.pdf`
10. Bitansky, N., Chiesa, A., Ishai, Y., Ostrovsky, R., Paneth, O.: Succinct non-interactive arguments via linear interactive proofs. In: Sahai, A. (ed.) Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings
11. Bitcoin SV: `https://github.com/bitcoin-sv/bitcoin-sv`
12. BitVM: Official BitVM implementation in Rust. `https://github.com/BitVM/BitVM`
13. Cairo: The Cairo Programming Language. `https://book.cairo-lang.org/title-page.html#the-cairo-book`
14. Ethereum Org: ZERO-KNOWLEDGE ROLLUPS. `https://ethereum.org/en/developers/docs/scaling/zk-rollups/`
15. Etherscan: etherscan.io. `https://etherscan.io/`
16. Firo: `https://firo.org`
17. franchfrog42: zkBaguette applied to zkSnarks. `https://github.com/frenchfrog42/zk-hackaton`
18. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct nizks without pcps. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings
19. Groth, J.: On the size of pairing-based non-interactive arguments. In: Advances in Cryptology - EUROCRYPT 2016
20. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Annual international cryptology conference. pp. 543–571. Springer (2016)
21. LumiBit: LumiBit's ZK-EVM. `https://lumibit.gitbook.io/lumibit-gitbook/overview/lumibit-101/lumibits-zk-evm`
22. Matsuda, Seiichi and Kanayama, Naoki and Hess, Florian and Okamoto, Eiji: Optimised Versions of the Ate and Twisted Ate Pairings. In: Cryptography and Coding. pp. 302–312. Springer Berlin Heidelberg (2007)
23. Merlin Chain: ZK Rollup on Bitcoin. `https://docs.merlinchain.io/merlin-docs/zk-rollup-on-bitcoin`
24. Miller, V.S.: The weil pairing, and its efficient calculation. Journal of Cryptology **17**(4), 235–261 (Sep 2004). `https://doi.org/10.1007/s00145-004-0315-8`
25. Patrick Fromberg: BSV Consensus Limits. `https://github.com/bitcoin-sv/bitcoin-sv/wiki/Consensus-Limits`

26. R. Linus: BitVM: Computing Anything on Bitcoin. https://bitvm.org/bitvm.pdf
27. Robin Linus and Lukas George: ZeroSync: Introducing Validity Proofs to Bitcoin. https://zerosync.org/zerosync.pdf
28. S. Nakamoto: Bitcoin: A Peer-to-Peer Electronic Cash System
29. Scott, M.: Pairing implementation revisited. Cryptology ePrint Archive, Paper 2019/077 (2019), https://eprint.iacr.org/2019/077, https://eprint.iacr.org/2019/077
30. Scott, M.: A note on twists for pairing friendly curves. http://indigo.ie/~mscott/twists.pdf
31. sCrypt Inc: https://scrypt.io
32. sCrypt Inc: sCrypt Transaction. https://test.whatsonchain.com/tx/2396a4e52555cdc29795db281d17de423697bd5cbabbcb756cb14cea8e947235
33. sCrypt Inc: Tutorial 5: Zero Knowledge Proofs. hhttps://docs.scrypt.io/tutorials/zkp
34. Sean Bowe: https://github.com/zkcrypto/jubjub
35. TAAL: whatsonchain.com. https://whatsonchain.com/
36. zCash: https://z.cash

Homomorphic Encryption Based ECDSA Generation Over Five Party Protocol

Akshit Aggarwal, Srinibas Swain

Department of Computer Science and Engineering, Indian Institute of Information Technology Guwahati, India.
{akshit.aggarwal, srinibas}@iiitg.ac.in

Abstract. Secure computation of multiparty protocol (MPP) is being widely used to address privacy issues in various technologies, such as cryptocurrencies, blockchain, and many more. The security of MPP is computed in various steps, mainly via secret sharing of keys, encryption/decryption of messages, and signature generation. However, literature suggests the secure computation of multiparty is arduous in the presence of adversarial nature of parties. In this paper, we propose a notion for achieving ECDSA in a multiparty protocol using an Authentication server A_t. We demonstrate this using four steps: 1. proposal of multiparty (five) protocol (5PP) in which there are parties with adversarial natures. 2. setup of 5PP with A_t that ensures the mutual identification of parties. 3. Encryption of messages in the proposed 5PP. 4. ECDSA signature generation between the parties in the aforementioned setup. We ensure security in all the steps mainly by using modified Paillier cryptosystem and homomorphic encryption.

Keywords: ECDSA · elliptic curve · five-party protocol · full secrecy · Homomorphic encryption · modified Paillier cryptosystem · ZkPok.

1 Introduction

Secure communication over multiparty protocol (MPP) is essential to preserve the privacy of each party within the network during message passing. Multiparty protocol enables n parties to ensure security of a system in the presence of x adversarial parties. The more number of adversarial parties can have an adverse affect on the system. Secure computation of parties can be maintained by a MPP (number of parties ≥ 5) that contains at least 2 adversarial parties [1]. The main contribution of this paper is to generate digital signature using ECDSA to ensure authenticity in a multiparty system. We achieve this in four stages.
In the first stage, we setup a multiparty system (where parties can be adversarial in nature) using five parties. Note that a multiparty system cannot be established in our context with less than five parties [1]. In the second stage, we use the notion of Authentication server A_t introduced by Tsai et al. [2] (as Authentication server ensures the mutual identification of parties). A_t stores the information of all parties. Parties interact with A_t by non-secret keys that helps to verify the

identity of users. Latter, in the third stage, we ensure the security of messages in 5PP by using modified Paillier cryptosystem [3]. Finally, we achieve full secrecy (where full secrecy is defined as when no intruder gets the information during the message transmission) in 5PP over a network by using homomorphic encryption. Homomorphic encryption (H_{em}) hides the information of sender and receiver over a network with the help of arbitrary analytic operation [4]. We also show that using elliptic curve cryptography (ECC), one can achieve a high level of security compared to RSA [5]. This implies a more secure system than RSA can be designed using a relatively smaller key by using ECDSA [6]. We first discuss the relevant literature for this study. ECDSA is helpful for maintaining authentic communication between the parties [7]. ECDSA can be mathematically expressed as:

$$\Omega = \alpha^{-1}(h(m) + \delta\gamma) \tag{1}$$

Here $h(m)$ is a hash function, m is a message, δ is a secret key, α is a secret nonce, and γ is public nonce (such that the x-coordinate of $\alpha \cdot \beta$ where β is base point generator of the elliptic curve).

Literature suggests that a limitation of ECDSA signature is that it does not maintain secrecy, mutual identification of parties, and the case when the number of parties are high. Also, ECDSA generation is quite expensive. Chien et al. [2] have proposed the ECC-based blind signcryption to strengthen the high level of security. The main drawback of their work is that they cannot achieve full secrecy. Xu et al. [8] used the signature key by threshold using the concept of Fiat-Shamir paradigm [9]. The main limitation is that their work leaks the information of both the sender and the receiver. Further, a few of the authors have extended the existing work of Xu et al. [8] by considering the Diffie-Hellman native assumptions and homomorphic Paillier encryption [3,10,11,12]. The main limitation of their works is that they did not discuss the adversarial nature of any party. Later on, a few authors extended the existing work by considering the five-party protocol with the adversarial nature of parties [13]. The main drawback is that they did not discuss the mutual identification between the parties. Afterwards, a few authors [14,15,16] proposed mutual identification based scheme to check the authentication of parties by considering trusted third party protocol. The work of [14] used identity based encryption to establish the mutual identification where key is distributed between multiple parties. The main limitation is high computation time for key agreement procedure. Li et al. [15] proposed third party based approach to mutually authenticate the system. The main limitation of their work is dishonest majority when the number of parties are scale up (by considering dishonest parties). Similarly, Sucasas et al. [16] proposed certificate authority (which provides some cryptographic keys to multi parties) to establish mutual identification. The main limitation of their work is complex protocol designing which increases the computation overhead over the parties. Further, Fragkos et al. [17] proposed several artificial intelligence based approach for authentication and encryption of message having limitation of assumed multiple trusted third party protocol which is not mutually identified. Further, Haung et al. [18] have proposed a partial blind ECDSA scheme widely used in bitcoins.

The main drawback of their work is that they use the Zero-knowledge Proof of knowledge (ZkPok), which increases the computation complexity. Xue et al. [19] have proposed two-party ECDSA signature generation via re-sharing of the secret key and linear sharing of the nonce. The main limitation of their work is that it does not applicable to more than two parties. Further, Ma et al. [20] have proposed linear secret sharing that does not require recursive operations for access control over message passing. The main limitation is that their work does not maintain full secrecy. Later, Aggarwal et al. [21] proposed homomorphic encryption over message passing with the help of ECDSA signature that maintains full secrecy. The main drawback is that they did not discuss the mutual identification of parties or the case when number of parties are more than two. Further, Aggarwal et al. [22] extended the work of [21] by considering the adversarial nature of parties using the claim of Byzantine agreement but they did not address the mutual identification between the parties.
In summary, we use 5PP as a multiparty protocol that contains adversarial parties. The communication between the parties is maintained by modified Paillier cryptosystem. The modified Paillier cryptosystem efficiently performs the homomorphic operations that provides security to the system as compared to other existing techniques [23,24]. Later on, this work establishes authentic communication with the help of ECDSA signature. ECDSA signature is helpful for generating the authentication in multiparty protocol.
The roadmap for the paper is as follows. Section 2 discusses the preliminaries. Section 3 discusses the proposed methodology. Security of the proposed work is discussed in Section 4. Finally, Section 5 discusses conclusion and future work.

2 Preliminaries

2.1 Elliptic Curve Cryptography (ECC)

In this work, message passing is performed via elliptic curves (EC). ECC was first introduced by Neal Kobiltz and Victor Miller in 1985 [25]. EC is a non-singular curve defined over a finite field Z_q (where q is the order of EC). EC consists of a set of points that satisfy the following equation, such as:

$$Y^2 = X^3 + AX + B \tag{2}$$

Where $X, Y, A, B \in Z_q$, and $4A^3 + 27B^2 \neq 0$.

2.2 Modified Paillier Cryptosystem

The encryption mechanism of this work is based on the modified Paillier cryptosystem, which is taken from the existing work of Xun et al. [3]. The modified Paillier cryptosystem is based on a large sample space, which is helpful for the easy computation of messages.
Consider a large prime number s (which is supposed to be a public parameter of ECDSA). The modified Paillier cryptosystem is discussed as follows.

– Key Generation: Randomly large prime numbers A and B are selected by parties such that the greatest common divisor $gcd(A-1, s) = gcd(B-1, s) = 1$. Afterwards, parties estimate:

$$N = AsB \tag{3}$$

$$M = (1+N)^{AB}(mod\ N^2) \tag{4}$$

, and publishes N, and M as public key keeping (A, B) as secret.
– Encryption: Encryption of message m is performed via randomly selecting I (where $m < q$ and $I \in Z_{N^2}$). Parties compute the ciphertext (CP) as:

$$CP = M^m I^N (mod\ N^2) \tag{5}$$

Here I is randomly selected, so the ciphertext looks like a random number.
– Decryption: The decryption (DC) of message is performed via secret key (A,B) such as:

$$DC = CP^{(A-1)(s-1)(B-1)}(mod\ N^2) \tag{6}$$

and message m can be computed as follows.

$$m = \left[\frac{DC-1}{ABN}\right][(A-1)(s-1)(B-1)]^{-1}(mod\ q) \tag{7}$$

2.3 Elliptic Curve Digital Signature Algorithm (ECDSA)

This section highlights the signature generation with the help of elliptic curves. EC plays a crucial role in achieving a high level of security with a smaller key size. The authenticity of messages over EC can be obtained by secret sharing of public key Q, secret key δ, public nonce γ, secret nonce α, and hash function $h(m)$ (where m is a message). EC is defined over a finite field Z_q consisting of an origin generator β such as:

$$Q = \delta\beta \tag{8}$$

,and

$$\gamma = \alpha\beta \tag{9}$$

Here β is also considered as x-coordinate $Q = \delta\beta$ (due to the symmetric properties of elliptic curves with respect to x-axis). Signature (Ω) over message with the help of EC is termed ECDSA, which can be mathematically expressed as:

$$\Omega = \alpha^{-1}(h(m) + \delta\gamma) \tag{10}$$

3 Proposed Methodology

This section indicates the methodology of our proposed work (as shown in Figure 1). This work is divided into four steps. These steps are essential for generating the ECDSA signature on the messages. The steps are discussed as follows.

1. Five party protocol (5PP)

$h^* = 1$
$x = 1$
$2x + h^* < n$
Here $n = 5$

2. Setup and Registration phase

P_i	P_j
$\delta_i \leftarrow Z_q$, $D_i \leftarrow Z_q$, $\alpha_i \leftarrow Z_q$	$\delta_j \leftarrow Z_q$, $D_j \leftarrow Z_q$, $\alpha_j \leftarrow Z_q$
$Q_i = \delta_i \beta$	$Q_j = \delta_j \beta$
$\pi_i = h_i(iv_i, Q_i)$	$\pi_j = h_j(iv_j, Q_j)$
$D_i = \alpha_i \beta = (x_{D_i}, y_{D_i})$	$D_j = \alpha_j \beta = (x_{D_j}, y_{D_j})$
$CF_i = \alpha_i^{-1}(\pi_i + x_{D_i}\delta_i)$	$CF_j = \alpha_j^{-1}(\pi_j + x_{D_j}\delta_j)$

3. Encryption (modified paillier cryptosystem)

P_i	P_j
$A \leftarrow Z_{N^2}$	$A \leftarrow Z_{N^2}$
$B \leftarrow Z_{N^2}$	$B \leftarrow Z_{N^2}$
$s \leftarrow Z_{N^2}$	$s \leftarrow Z_{N^2}$
$N = AsB$	$N = AsB$
$M = (1+N)^{AB} \pmod{N^2}$	$M = (1+N)^{AB} \pmod{N^2}$

4. ECDSA Signature Generation

4.1. Homomorphic Encryption

P_i	P_j
$\delta_i \leftarrow Z_q$	$\delta_j \leftarrow Z_q$
$I_i \leftarrow Z_{N^2}$	$\alpha_j \leftarrow Z_q$
	$\gamma_j = \alpha_j \gamma$
	$I_j \leftarrow Z_{N^2}$
Encrypt with hash function $h(m_i)$	Encrypt with public nonce γ with respect to secret key δ
$CP_i = M^{h(m_i)} I_i^N \pmod{N^2}$	$CP_j = (M^{\gamma_j} I_j^N)^{\delta_j} \pmod{N^2}$

4.1.1. Generation of CP_o

$CP_o = H_{em}[(CP_1, CP_2), *]$
$CP_o = M^{(h(m_i)+\gamma_j\delta_j)} I_i^N I_j^{N\delta_j} \pmod{N^2}$

4.2 Signature generation

$$\Omega = \alpha^{-1}(h(m) + \gamma\delta)$$

Fig. 1. Flow diagram of ECDSA generation over five-party protocol (5PP). Here parties P_i and P_j are randomly selected from 5PP (where $1 \leq i, j \leq 5$, and $i \neq j$).

3.1 Multiparty (five) Protocol (5PP) with adversarial nature

In this step, for maintaining secure communication, multiparty (where number of parties ≥ 5) protocol with an adversarial nature is used, which was adopted from the work of Alon et al. [1]. Here we remark that the system is not applicable when the number of parties are less than five. They worked on n-party system with an honest majority, which can be mathematically expressed as:

$$2x + h^* < n \tag{11}$$

Here n is total number of parties. h^* is number of semi-honest parties (where a party is considered semi-honest if it follows all the protocol of the proposed network but is curious to get the information of other parties), and x is number of adversarial parties. For attaining minimum multiparty protocol, we assume $x = 1$ and $h^* = 1$, then the minimum possible value for $n = 4$. When $n = 4$, it is difficult for any party to ensure the system's security (as $n = 4$ disobeys the claim of Byzantine agreement [26]). For the sake of simplicity, we assume that $n = 5$ (say 5PP), which ensures the security of multiparty protocols. The proof for 5PP is discussed in Section 4.

3.2 Setup of 5PP with Authentication server A_t

After measuring adversarial nature of parties, this work uses the notion of an Authentication server, A_t. A_t stores information about all parties that helps check the mutual identification. The mutual identification of phases are discussed as follows.

- Firstly, each party computes the value of its public key, such as:

$$\begin{aligned} Q_1 &= \delta_1 \beta \\ Q_2 &= \delta_2 \beta \\ Q_3 &= \delta_3 \beta \\ Q_4 &= \delta_4 \beta \\ Q_5 &= \delta_5 \beta \end{aligned} \tag{12}$$

  Here $\{\{Q_1, Q_2, \cdots, Q_5\} \in Z_q\}$ are public keys, $\{\{\delta_1, \delta_2, \cdots, \delta_5\} \in Z_q\}$ are secret keys, β is base point generator of elliptic curve for parties $\{P_1, P_2, \cdots, P_5\}$.
- After computing public keys, each party sends its information to the A_t with secret keys and their unique identity values (iv). Then, A_t applies hash function ($h(\cdot)$) and generates the non-secret salt value, π, for verifying identity of each participant. The corresponding hash value is obtained as follows:

$$\begin{aligned} \pi_1 &= h_1(iv_1, Q_1) \\ \pi_2 &= h_2(iv_2, Q_2) \\ \pi_3 &= h_3(iv_3, Q_3) \\ \pi_4 &= h_4(iv_4, Q_4) \\ \pi_5 &= h_5(iv_5, Q_5) \end{aligned} \tag{13}$$

Here $\{\{\pi_1, \pi_2, \cdots, \pi_5\} \in Z_q\}$ are non-secret salt values, $\{h_1, h_2, \cdots, h_5\}$ are corresponding hash values with unique identity values. $\{iv_1, iv_2, \cdots, iv_5\}$ for parties $\{P_1, P_2, \cdots, P_5\}$.

– Each party has their own data points, such as $\{\{D_1, D_2, \cdots, D_5\} \in Z_q\}$ on the elliptic curve, with β being the base point generator. Each party selects their secret nonce values $\{\{\alpha_1.\alpha_2, \cdots, \alpha_5\} \in Z_q\}$ and computes the value of data points such as:

$$\begin{aligned} D_1 &= \alpha_1\beta = (x_{D_1}, y_{D_1}) \\ D_2 &= \alpha_2\beta = (x_{D_2}, y_{D_2}) \\ D_3 &= \alpha_3\beta = (x_{D_3}, y_{D_3}) \\ D_4 &= \alpha_4\beta = (x_{D_4}, y_{D_4}) \\ D_5 &= \alpha_5\beta = (x_{D_5}, y_{D_5}) \end{aligned} \tag{14}$$

Here x_{D_i} and y_{D_i} (where $i \in \{1, 2, \cdots, 5\}$) are the coordinates of x-axis and y-axis corresponding to the data points.

– After computing the data points, each party P_i (where $i \in \{1, 2, \cdots, 5\}$) generates the certificates (CF_i) with the help of A_t such as:

$$\begin{aligned} CF_1 &= \alpha_1^{-1}(\pi_1 + x_{D_1}\delta_1) \\ CF_2 &= \alpha_2^{-1}(\pi_2 + x_{D_2}\delta_2) \\ CF_3 &= \alpha_3^{-1}(\pi_3 + x_{D_3}\delta_3) \\ CF_4 &= \alpha_4^{-1}(\pi_4 + x_{D_4}\delta_4) \\ CF_5 &= \alpha_5^{-1}(\pi_5 + x_{D_5}\delta_5) \end{aligned} \tag{15}$$

Later on, the A_t computes the certificate value in terms of binary output, that is, 0 or 1. If the obtained value is 1 means party is valid, else for 0 party either is malicious or semi-honest. The corresponding value of certificates helps obtain the nature of the parties. This technique maintains mutual identification among the parties.

3.3 Modified Paillier Encryption

A Paillier cryptosystem is efficient for performing homomorphic encryption, which maintains full secrecy over the network [27]. In this work, we apply the modified Paillier cryptosystem as it is more secure than the Paillier cryptosystem, which is taken from the existing work of Xun et al. [3]. Here parties P_i and P_j are randomly selected (where $i, j \in \{1, 2, \cdots, 5\}$ and $i \neq j$), which works as a sender and a receiver in the encryption mechanism of messages.
As we know, $M_i = (1 + N)^{AB}(mod\ N^2)$, then

$$M_i^s = (1 + N)^{AsB} = (1 + N)^N = 1 + N^2 = 1\ (\ mod\ N^2).$$

Therefore,

$$\begin{aligned} DC_i &= CP_i^{(A-1)(s-1)(B-1)} \ (mod\ N^2) \\ &= M^{m_i(A-1)(s-1)(B-1)} \alpha^{N(A-1)(s-1)(B-1)} \\ &= (1+N)^{AB[m_i(A-1)(s-1)(B-1)\ (mod\ s)]} \\ &\alpha^{N(A-1)(s-1)(B-1)} \\ &= 1 + AB[m_i(A-1)(s-1)(B-1)\ (mod\ s)]N \end{aligned}$$

Here $AB[m_i(A-1)(s-1)(B-1)\ (mod\ s)] < N$. According to Euler theorem, we know for any non-zero integer χ (where $\chi \in Z_{N^2}$), where $\chi^{\phi(N^2)} = 1\ (mod\ N^2)$, and $\chi \in Z_{N^2}$ [28].
In the competence of this, we have:

$$AB[m_i(A-1)(s-1)(B-1)\ (mod\ s)] = (DC_i - 1)/N$$

Therefore, we have:

$$m_i = \left[\frac{DC_i - 1}{ABN}\right] [(A-1)(s-1)(B-1)]^{-1} (mod\ s) \tag{16}$$

3.4 ECDSA signature generation

The generation of ECDSA is formed in three steps. Firstly, we apply the ZkPok for the reduction of complexity. In the second step, we apply homomorphic encryption (which is helpful for maintaining secrecy over the network). Afterward, the signature is generated to check the authenticity of messages.

Zero knowledge Proof of Knowledge (ZkPok) As we know, the value of public key Q and public nonce γ can be computed as $Q = \delta\beta$, and $\gamma = \alpha\beta$. Computing the value of δ and α by the dint of ZkPok is quite arduous. In this proposed work, we compute the values of δ and α with the help of non-interactive Zero-knowledge Proof of Knowledge ($niZkPok$) using the concept of Fiat-Shamir paradigm [9,29]. This step reduces the computation of the proposed work.

Homomorphic Encryption In homomorphic encryption, message stays encrypted at all times, which minimises the likelihood that message gets compromised. In this step, parties P_i and P_j are randomly selected (where $i \neq j$). Afterwards, party P_i encrypts the message m_i with respect to hash function $h(\cdot)$, such as:

$$CP_i = M^{h(m_i)} I_i^N (\bmod\ N^2) \tag{17}$$

Later on, party P_j encrypts the message with public nonce γ and secret key δ_j, such as:

$$CP_j = (M^{\gamma_j} I_j^N)^{\delta_j} (\bmod\ N^2) \tag{18}$$

Further, CP_i and CP_j homomorphically encrypt (H_{em}) messages with the help of analytical function *i.e.*, multiplication ($*$) and generate the ciphertext as follows.

$$\begin{aligned} CP_o &= H_{em}[(CP_1, CP_2), *] \\ &= M^{h(m_i)} I_i^N \ (mod\ N^2) * (M^{\gamma_j} I_j^N)^{\delta_j} \ (mod\ N^2) \\ &= M^{(h(m_i)+\gamma_j\delta_j)} I_i^N I_j^{N\delta_j} (mod\ N^2) \end{aligned} \tag{19}$$

Signature Generation After getting CP_o, the receiver estimates the signature (Ω) with the help of secret random value (α^{-1}) (where α^{-1} is the inverse of secret random value; as we know, the inverse operation is quite difficult for computation) such as:

$$\Omega = \alpha^{-1}(h(m_i + \gamma_j\delta_j) \tag{20}$$

Here, we remark that during the signature generation time for a particular party, we assume that $i = j = 1$. Then, from Equation 20, the signature can be computed as follows.

$$\Omega = \alpha^{-1}(h(m_1) + \gamma_1\delta_1)$$

The optimised version for the signature can be expressed as:

$$\Omega = \alpha^{-1}(h(m) + \gamma\delta) \tag{21}$$

Finally, receiver receives signature as (δ, Ω).

4 Security of proposed work

In this section, we compute the security of our proposed work. In this paper, security is computed in various steps. The steps are discussed as follows.

4.1 Five party protocol (5PP)

As from the existing work of Alon et al. [1], $2x + h^* < n$. If we consider the value of $x = 1$ and $h^* = 1$, then the least value of $n = 4$. As the value of $n = 4$, the system is not secure. At $n = 4$, a system has four parties (say, P_1, P_2, P_3, and P_4). These four-party protocols consist of message m (where party P_1 is randomly selected as a malicious party, party P_2 is selected as a semi-honest party, and the rest parties are considered honest). For ensuring security, we assume that any honest party (say P_4) receives m' from P_1 (as P_1 is malicious party), m'' from P_2 (as P_2 is semi-honest), and m from P_3. In that scenario, the selection of message is arduous for P_4.
For sake of simplicity, at $n = 5$, a system has five parties (say, P_1 (malicious party), P_2 (semi-honest party), P_3 (honest party), P_4 (honest party), and P_5 (honest party)) that consist of message m. For ensuring security, we assume that any honest party (say P_5) receives m' from P_1, m'' from P_2, and m from the

rest of parties (as all other parties are honest). In that scenario, party P_4 selects the honest majority by considering the claim of Byzantine agreement [26]. The claim of Byzantine agreement selects two-thirds honest majority that ensures the security of a multiparty protocol in the presence of adversarial behavior by parties. Hence, at $n = 4$, the system does not ensure security. So, in this work, five-party protocol (5PP) is applied, which ensures security even in the presence of adversarial parties.

4.2 Modified Paillier Cryptosystem

In this work, we prove the modified Paillier cryptosystem is secure by taking a random oracle (where the oracle always generates the output either Yes or No). We begin this problem by taking large composite residues. The heart of this proof lies in modulus where modulus is performed on squares. As from paillier encryption, we know that $N = AsB$ (where $A, s, B \in Z_{N^2}$).
Definition. An integer λ is said to be the N-th residue modulo N^2 if there exists an integer μ such that $\lambda = \mu^N (mod\ N^2)$.
The N-th residuosity is a multiplicative subgroup of Z_{N^2} of order s, such as:

$$\phi(N) = (A-1)(s-1)(B-1) \tag{22}$$

Here λ has exactly N-th roots of degree N (where exactly one root is strictly less than N). The N-th roots of unity can be mathematically expressed as:

$$(1+N)^z = 1 + zN(mod\ N^2), \tag{23}$$

(where $z \in Z_q$). Deciding the P-th residuosity is NP-hard. Accordingly, we assume that:

Conjecture. There exists no polynomial time solver for N-th residues modulo N^2 [Section 5, [3]].
As we know from Paillier encryption, $M = (1+N)^{AB}$. We denote it by $\rho(m, v)$ which can be defined as:

$$\rho(m, v) = M^m v^N (mod\ N^2) \tag{24}$$

(where $m \in Z_q$ and $v \in Z_{N^2}$). Assume $\omega = \rho(m, v) \in Z_{N^2}$.

Claim. Random oracle decides the security of modified Paillier cryptosystem when $z = m$.
As we know, $\omega = M^m v^N (mod\ N^2)$ and we can write it as:

$$\omega M^{-z} = M^{m-z} v^N (mod\ N^2) \tag{25}$$

that submits to the random oracle for solving the residue problem. In case of N-th residue, there exists a constant v'^N (where $v'^N \in Z_{N^2}$) such as:

$$\omega M^{-z} = v'^N (mod\ N^2) \tag{26}$$

From Equations (25, 26), we can deduce that:

$$M^{m-z} \quad \frac{v}{v'} \quad ^{N} = 1(mod\ N^2) \tag{27}$$

(where v, v' are constants). So by raising power of $(A-1)(s-1)(B-1)$ on both sides, we can deduce that:

$$M^{(m-z)(A-1)(s-1)(B-1)(mod\ s)} = 1(mod\ N^2) \tag{28}$$

Further, we can compute it is as:

$$(m-z)(A-1)(s-1)(B-1)(mod\ s) < s \tag{29}$$

Now, consider the claim when $z = m$ Equation (29) holds good. We can conclude that random oracle solves the problem and provides the outcome Yes. In the above competency, we can deduce that the modified Paillier cryptosystem is secure.

4.3 Elliptic Curve Discrete Logarithmic Problem (ECDLP)

In this step, security is computed in terms of ECDLP [30]. As from Equations (8, 9), $Q = \delta\beta$, and $\gamma = \alpha\beta$.
Computation of Q and γ is adequate with the knowledge of δ, α, and β. Consider a situation where an intruder knows the value of Q, γ, and β. Due to inadequate knowledge, it is arduous to compute the value of δ and α. Hence, the discrete logarithmic problem of elliptic curves improves the security of system.

4.4 Homomorphic Encryption

In this work, we homomorphically encrypt the ciphertexts of parties P_i and P_j with the multiplication operation. As we know from Equation (19):

$$CP_o = H_{em}[(CP_1, CP_2), *] \tag{30}$$

Consider a situation where, at any instance, when an attacker gets the information of CP_1 and CP_2 with respect to CP_o, then due to inadequate knowledge of arbitrary operation, that is, multiplication $(*)$, an attacker cannot get the information of CP_o. Thus, homomorphic encryption makes the proposed work secure.

4.5 Signature Generation

Finally, for the signature generation, we use the value of secret key (α). We are proposing the inverse operation of secret key, which is quite difficult to compute. Hence, inverse operations increase the security of the system (as message is always encrypted throughout the communication channel).

5 Limitations

This work does not verify the designed protocol concerning side channels attacks (or any cryptographic attacks). The proposed work does not discuss the implementation setup.

6 Conclusion and Future work

In this work, homomorphic encryption-based ECDSA generation over five-party protocol is proposed. We show the mutual identification between the parties with the help of A_t, and the security of messages is ensured by modified Paillier cryptosystem. Finally, our work achieves the authenticity of 5PP with the help of ECDSA signature.
In future, we will extend our work to more than five parties. Another extension of this study is to maintain full secrecy between the adversarial parties.

References

1. B. Alon, E. Omri, and A. P. Cherniavsky, "MPC with Friends and Foes," in *Proceedings of the 40th Annual International Cryptology Conference*, 2020, pp. 677–706.
2. C. H. Tsai and P. C. Su, "An ECC-Based Blind Signcryption Scheme for Multiple Digital Documents," *Security and Communication Networks*, pp. 1–14, 2017.
3. X. Yi and K. Y. Lam, "A New Blind ECDSA Scheme for Bitcoin Transaction Anonymity," in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, 2019, pp. 613–620.
4. K. K. Chauhan, A. K. S. Sanger, and A. Verma, "Homomorphic Encryption for Data Security in Cloud Computing," in *Proceedings of 2015 International Conference on Information Technology (ICIT)*, 2015, pp. 206–209.
5. M. A. Saadi and B. Kumar, "A Review on Elliptic Curve Cryptography," *International Journal of Future Generation Communication and Networking*, vol. 13, no. 2, pp. 1597–1601, 2020.
6. A. Roy and S. Karforma, "A survey on digital signatures and its applications," *Journal of Computer and Information Technology*, vol. 3, no. 1, pp. 45–69, 2021.
7. D. Johnson, A. Menezes, and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.
8. Z. Xu, Y. Pan, and Z. Tao, "A Threshold Signature Key Protection Scheme Based on Blind Technology," in *Proceedings of the 2017 International Conference on Information Technology*, 2017, pp. 157–161.
9. A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO*, 1986, pp. 186–194.
10. J. Doerner, Y. Kondi, E. Lee, and A. Shelat, "Secure Two-party Threshold ECDSA from ECDSA Assumptions," in *Proceedings of 2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 980–997.
11. R. Gennaro and S. Goldfeder, "Fast Multiparty Threshold ECDSA with Fast Trustless Setup," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 1179–1194.

12. G. Castagnos, D. Catalano, F. Laguillaumie, F. Savasta, and I. Tucker, "Bandwidth-Efficient Threshold EC-DSA," in *Proceedings of 23rd IACR International Conference on Public-Key Cryptography*, 2020, pp. 266–296.
13. N. Koti, V. B. Kukkala, A. Patra, and B. R. Gopal, "PentaGOD: Stepping Beyond Traditional GOD with Five Parties," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1843–1856.
14. A. K. Sahu and S. Sharma, "Lightweight Multi-party Authentication and key Agreement Protocol in IoT-based E-Healthcare Service," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 64, pp. 1–20, 2021.
15. X. Li, K. Zhang, L. Zhang, and X. Zhao, "A New Quantum Multiparty Simultaneous Identity Authentication Protocol with the Classical Third-Party," *Entropy*, vol. 24, no. 4, pp. 1–10, 2022.
16. V. Sucasas, A. Aly, G. Mantas, J. Rodriguez, and N. Aaraj, "Secure Multi-Party Computation-Based Privacy-Preserving Authentication for Smart Cities," *IEEE Transactions on Cloud Computing*, vol. 11, no. 4, pp. 3555–3572, 2023.
17. G. Fragkos, J. Plusquellic, C. Minwalla, and E. E. Tsiropoulou, "Artificially Intelligent Electronic Money," in *IEEE Consumer Electronics Magazine*, vol. 10, no. 4, 2020, pp. 81–89.
18. H. Haung, Z. Y. Liu, and R. Tso, "Partially Blind ECDSA Scheme and Its Application to Bitcoin," in *Proceedings of the 2021 IEEE Conference on Dependable and Secure Computing (DSC)*, 2021, pp. 1–8.
19. H. Xue, M. H. Au, X. Xie, T. H. Yuen, and H. Cui, "Efficient Online-friendly Two-Party ECDSA Signature," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 558–573.
20. R. Ma and L. Du, "Attribute-Based Blind Signature Scheme Based on Elliptic Curve Cryptography," *IEEE Access*, vol. 10, pp. 34 221–34 227, 2022.
21. A. Aggarwal and S. Swain, "Blind Two Party ECDSA Signing Based Homomorphic Encryption over Message Passing," in *IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA)*, 2022, pp. 1–5.
22. ——, "Poster: Correctness of n-parties ECDSA By the Claim of Byzantine Agreement," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3319–3321.
23. Q. ShenTu and J. Yu, "A Blind-Mixing Scheme for Bitcoin based on an Elliptic Curve Cryptography Blind Digital Signature Algorithm," *arXiv preprint arXiv:1510.05833*, 2015.
24. F. Knirsch, A. Unterweger, M. Unterrainer, and D. Engel, "Comparison of the Paillier and ElGamal Cryptosystems for Smart Grid Aggregation Protocols," in *ICISSP*, 2020, pp. 232–239.
25. N. Kobiltz, "Elliptic curve cryptosystem," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
26. L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals," *ACM Transaction on Programming Languages and Systems*, vol. 4, no. 3, pp. 382–401, 1982.
27. N. Fazio, R. Gennaro, T. Jafarikhah, and W. E. Skeith, "Homomorphic Secret Sharing from Paillier Encryption," in *International Conference on Provable Security*, 2017, pp. 381–399.
28. M. Laššák and Š. Porubský, "Fermat-Euler Theorem in Algebraic Number Fields," *Journal of Number Theory*, vol. 60, no. 2, pp. 245–290, 1996.
29. M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *STOC*, 1988, pp. 103–112.

30. C. Diem, "On the discrete logarithm problem in elliptic curves," *Compositio Mathematica*, vol. 147, no. 1, pp. 75–104, 2011.

Benchmarking post-quantum cryptography in Ethereum-based blockchains

Patxi Juaristi[1], Isaac Agudo[1], Ruben Rios[1], Laura Ricci [2]

[1] NICS Lab, Universidad de Málaga, Spain
[2] Università di Pisa, Italy

Abstract. Blockchain technology has significantly transformed various industries by enabling secure and tamper-resistant transactions. However, the rise of quantum computing threatens the cryptographic foundations of blockchain networks, making blockchain vulnerable to signature forgery and transaction manipulation. This raises concerns about the long-term viability of blockchain systems and highlights the need for post-quantum secure solutions.
This paper investigates the feasibility of quantum-resistant blockchain ecosystems. Our research focuses on estimating the cost of the integration of the post-quantum algorithms selected in the NIST standardization competition into Ethereum-based blockchains.

Keywords: Blockchain · post-quantum cryptography · digital signature · Ethereum · ECDSA · Dilithium · Falcon · SPHINCS$^+$.

1 Introduction

In recent years, blockchain technology has emerged as a revolutionary technology for secure and decentralized data management, largely due to its ability to provide an immutable and transparent data ledger, which cannot be manipulated. This technology has been adopted in various sectors, with finance being the main application scenario, but also extending to supply chain management and many others.

However, the advent of quantum computers poses a major threat to the foundations of blockchain technology as they could undermine the security of current cryptographic algorithms. Leveraging the principles of quantum mechanics, quantum computers can perform a large number of calculations simultaneously because their basic unit of information representation, the quantum bit or qubit, can exist in a superposition of states. This allows multiple states to be represented at the same time, greatly facilitating efficient parallel processing [1].

As such, quantum computers are capable of solving complex mathematical problems considerably faster than classical computers.

The potential of quantum computing is significantly enhanced by the application of specific algorithms, such as Shor's algorithm [2] for factoring large numbers, and Grover's algorithm [18] for speeding up searches in unstructured data.

These algorithms have direct implications for the security of current cryptographic algorithms, especially public-key algorithms. For example, the security of RSA is based on the impossibility of factoring large prime numbers, but with Shor's algorithm a 2048-bit RSA key could be factorized in approximately 8 hours using a quantum computer with 20 million qubits [13].

While this is a clear threat to security, the reality is that we are still far from building quantum computers with this number of qubits. To the best of our knowledge, the largest quantum computers to date have just over a thousand of qubits. IBM's Condor, for example, has 1.121 qubits [14]. On such a computer, the factorization of the above mentioned RSA key would take approximately 142.204 hours, or about 5915 days. This value was obtained without differentiating between physical and logical qubits, because otherwise the result would be much higher [3].

Although quantum computers are not an immediate problem, the rapid development of them in recent years, has led to the need for new cryptographic algorithms that can withstand quantum computers. These new cryptographic algorithms are referred to as post-quantum or quantum-resistant algorithms.

Since blockchains rely on public-key cryptography to operate, they are also vulnerable to quantum computers. The main mechanism for interacting with blockchains is through transactions, which are digitally signed to ensure their authenticity. Most blockchains, including Bitcoin and Ethereum-based blockchains, use the Elliptic Curve Digital Signature Algorithm (ECDSA) [15] for this purpose. The main reason is that ECDSA uses short keys and produces short signatures.

In this paper we investigate the threat that quantum computers pose to blockchain systems and whether post-quantum cryptography (PQC) can be integrated into Ethereum-based blockchains to mitigate the risk. The main contribution of this paper is a performance comparison of the PQC algorithms selected from the NIST standarization process against ECDSA using real-time transaction data. It has been focused only on the ECDSA signature algorithm used to sign the Ethereum transactions, not in the BLS signature. The reason for this has been because replacing BLS is much more changeling given that Ethereum consensus uses BLS signature aggregation to store the results of the consensus voting. In the end, the solution has been composed of a modular and scalable system for acquiring, comparing and visualizing the results.

The rest of this paper is organized as follows. Section 2 provides a comparative analysis with related works. Next, Section 3 introduces the main existing post quantum cryptography families. The benchmarking architecture is described in Section 4. Subsequently, Section 5 shows the results obtained from the evaluation of applying ECDSA and NIST selected algorithms to real-time transaction data from an actual blockchain network. Finally, Section 6 present the conclusion and outlines potential lines of future research.

2 Related work

Concerns about the threat of quantum computers to current cryptographic schemes have led to notable research in this area, generating several solutions and surveys.

Regarding survey papers, Buser et al. [4] focus on exotic signature schemes for post-quantum blockchains, exploring the challenges associated with their implementation and proposing research directions. More recently, Yang et al. [21] presented a comprehensive survey and comparison of post-quantum and quantum blockchains, which highlighted the current state of research and identifying possible future directions.

Some papers focus on investigating the application of post-quantum blockchains to particular scenarios. For example, Chen et al. [5] concentrate on studying the practical implications of integrating post-quantum cryptographic schemes into blockchain systems designed for the Internet of Things (IoT) and other smart city infrastructures. Similarly, Yi et al. [22], discuss the application of post-quantum blockchain technology to secure the social Internet of Things (SIoT), proposing a framework that leverages post-quantum cryptography to ensure data integrity and privacy in these scenarios.

Additionally, some authors proposed the use of lattice-based cryptography to make blockchain networks resistant to quantum attacks. The focus of [11] is the creation of a cryptocurrency based on a post-quantum blockchain while [16] proposes a new signature scheme and describes how to apply it to secure blockchain transactions. None of them present a practical implementation.

To the best of our knowledge, the paper that is most similar to ours is [9]. This paper analyses the feasibility of post-quantum algorithms for the blockchain. However, their study does not use the finalists of the PQC competition organized by NIST. Furthermore, their evaluation is not done with real blockchain data.

In contrast, in this paper we provide a modular and scalable tool to facilitate the incorporation of novel post-quantum algorithms as they are developed. In addition, our solution uses real-time transaction data from a blockchain network, making the evaluation results more accurate.

3 Post-quantum cryptography families

The threat of quantum computing to today's cryptographic schemes has prompted the US National Institute of Standards and Technology (NIST) to launch a standardization process for post-quantum schemes, which aims to develop and standardize cryptographic algorithms that are resistant to quantum attacks [17]. The competition started in 2016 with 69 candidate algorithms and has progressed through rounds of evaluation. The candidates algorithms can be classified into different families, which are briefly described below.

Lattice-based cryptography relies on the mathematical properties of lattices, geometric structures formed by points in n-dimensional space. It involves solving problems such as the Shortest Vector Problem (SVP) and the Closest

Vector Problem (CVP) in these high-dimensional spaces, making brute-force attacks impractical. Lattices are often represented as arrays for easier matrix operations. The main cryptographic schemes include NTRUEncrypt and NTRUSign, which leverage SVP and CVP, and Ring-LWE-based schemes, which use the Ring Learning with Errors (RLWE) problem to perform computations on polynomial rings.

Code-based cryptography is one of the oldest and most studied approaches to post-quantum cryptography. Its security is based on the difficulty of solving the mathematical problems associated with error-correcting codes. These codes ensure reliable data transmission over noisy channels by introducing redundancy, which allows the receiver to detect and correct errors, while attackers without secret knowledge cannot decode the data. The McEliece cryptosystem, created in 1978, is the most famous code-based algorithm. It encrypts messages by encoding them in codewords and adding random errors to make decryption impossible without the private key. Although efficient and secure, the McEliece cryptosystem has large public keys and computationally intensive key generation and management processes.

Multivariate polynomial-based cryptography (MPKC) is a cryptographic scheme that exploits the hardness of solving systems of multivariate polynomial equations, which are computationally very difficult to solve, and have been proven to be NP-Complete, like some other lattice or code problems. These problems are of complexity class NP (non-deterministic polynomial time), which means that if a polynomial-time algorithm exists for solving any NP-complete problem, then polynomial-time algorithms exist for solving all problems in NP, making them essentially equivalent in difficulty.

Hash-based cryptography is a method of encrypting and securing data using hash functions. Before the process begins, it is necessary to determine which values are to be signed, and then to generate a long random string of characters for each of them. This resulting random string will be the private key used to sign the data. Once the private key is ready, this string is hashed using typical hash functions such as SHA-1, SHA-3, SHA-256 or BLAKE2, which produce strings between 256 and 512 bits, which are used as public keys to verify the signature. This is done by hashing the signature (the private key) and comparing it with the public key it has.

Isogeny-based cryptography creates cryptographic systems over finite fields using isogeny graphs of elliptic curves, which are mappings that preserve algebraic structures. An isogeny is a morphism between two elliptic curves, characterized by its homomorphism between the groups of points on the curves. Key generation involves choosing two elliptic curves over a finite field and generating a secret isogeny between them. The private key is the isogeny, while the public key is made of the starting and resulting curves. While the Supersingular Isogeny Diffie-Hellman (SIDH) protocol and its successor, Supersingular Isogeny Key Encapsulation (SIKE), were initially prominent, they have been found vulnerable to cryptographic attacks. As a result, newer isogeny-based schemes like SQISign and its variants have emerged. SQISign, which has been featured in recent NIST

post-quantum cryptography evaluations, represents a more robust approach, addressing the vulnerabilities of earlier protocols and offering promising security in the evolving landscape of post-quantum cryptography.

4 Benchmarking solution

This section describes the benchmarking tool developed to evaluate the performance of both current and post-quantum cryptographic algorithms in blockchain.

The tool consists of a set of interconnected functional blocks that result in a modular and scalable environment for the evaluation of cryptographic algorithms against real-time transactions from an Ethereum-based blockchain. The tool also provides an interface for the visualization and analysis of the results.

In the following, we present the overall architecture of the benchmarking solution and its building blocks in detail.

4.1 System architecture

The proposed solution consists of four main components as illustrated in Figure 1. The first component is devoted to the deployment of a blockchain network. The second component provides all necessary cryptographic algorithms, which will be applied to actual blockchain data in the third component. The fourth component provides a mechanism to display the results.

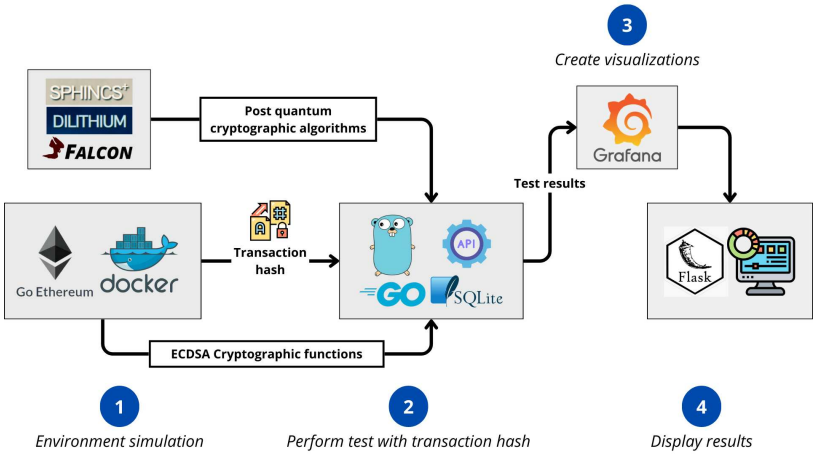


Fig. 1: Benchmarking solution architecture

As shown in the figure, our solution incorporates an actual blockchain network capable of deploying and running a number of Go-Ethereum nodes using a

Docker environment. All transactions sent in the blockchain environment are intercepted and their actual payloads are passed to our benchmarking API. The benchmarking component evaluates the performance of both the cryptographic algorithms currently provided by the Go-Ethereum client and the post-quantum cryptographic schemes selected by NIST. The results are stored in a Sqlite database that is read by Grafana, the component in charge of generating graphics for comparing the evaluation results. These graphics are finally included in a web application developed with Flask to facilitate the visualization and analysis of the results.

4.2 Cryptographic component

This section presents the cryptographic algorithms and libraries incorporated in our benchmarking solution. Currently, it includes the three post-quantum signature algorithms selected by NIST after their standardization competition:

- **CRYSTALS-Dilithium** [7]: Dilithium is a digital signature algorithm based on the hardness of lattice problems over module lattices. It ensures that even with access to a signing oracle, an adversary cannot fabricate a new signature for an unsigned message or a different signature for an already signed message. Dilithium utilizes rejection sampling of Fiat-Shamir with Aborts and the uniform distribution for signature generation, leading to secure yet larger signatures. Despite this, its key size is reduced post-creation with an optimizaiton process, making it the algorithm with the smallest combined public key and signature size among lattice-based schemes. Dilithium offers three modes (Dilithium 2, 3, and 5), with increasing security and resource needs, and includes AES encryption.
  As there is a library that implements this algorithm in GO [6], the necessary test functions have been included directly in the simulation program that has been carried out in GO.
- **Falcon** [10]: As Dilithium, it is a lattice-based digital signature scheme, but uses the Gentry, Peikert, and Vaikuntanathan framework [12], deploying NTRU lattices with a fast Fourier sampling trapdoor sampler. This scheme addresses the short integer solution (SIS) problem over NTRU lattices, which remains computationally challenging even for quantum computers. Falcon offers two modes, Falcon-512 and Falcon-1024, with this benchmark focusing on Falcon-1024 due to its higher security.
  To implement of Falcon, since there is currently no native library for GO, it was necessary to use the CGO library to combine C and GO code, calling Falcon cryptographic functions from C [19] for analysis in GO.
- **SPHINCS$^+$** [20]: Stateless hash-based digital signature scheme using Merkle tree structures. It offers high security with relatively short signatures and fast verification times. SPHINCS$^+$ includes three variants based on the underlying hash functions: SHAKE256, SHA-256, and Haraka, with this benchmark focusing on SHA-256.

The GO library for SPHINCS$^+$ [8] supports 12 modes per hashing method, varying in hash length (128, 192, 256) and mode type (simple, robust). In this case, six modes have been analyzed, covering both simple and robust options for each hashing type.

Although this component only incorporates libraries for the three finalists of the NIST competition, it is designed to facilitate the inclusion of new cryptographic algorithms in the future.

4.3 Measurement component

The measurement component is one of the core elements of the proposed solution. This component is responsible for receiving, in real time, the transactions of all blocks generated by the Ethereum-based blockchain.

Every time a signature is generated or verified in the blockchain network, an API call is made with the hash of the transaction. This hash is then used as payload to perform the comparisons of cryptographic functions.

The tests, which are performed in every single API call, simulate a key generation, signature generation and signature verification process. These tests are first carried out with the cryptographic code extracted from Go-Ethereum, i.e. with ECDSA, and then compared with the post-quantum cryptographic schemes explained above: Dilithium, Falcon and SPHINCS$^+$.

The metrics measured have been the execution time and the memory usage of the corresponding cryptographic function. All measurements have been carried out directly from GO, with native libraries: `time` for time measurement and `runtime/pprof` for CPU profiling. Both are started just before calling the function to which the parameters are going to be measured.

However, instead of measuring the time directly in the blockchain node, we produce also an ECDSA signature in the backend to ensure that all benchmarked algorithms are run in the same environment. As the backend does not know the private keys used for signatures, we need to generate a new pair in each test iteration. In addition to that, key and signature sizes have also been measured apart from the tests.

Execution times The simulations have been carried out several times in a loop and the average execution time of all of them has been calculated, since there have been functions that took extremely low time that were considered as 0ms by the measurement function. This loop was 50 times for the tests performed by API call.

However, the need to perform iterations to obtain more accurate results, increased considerably the time required to complete each test. This has caused another problem, which was that API calls were being received with new block information, while tests were still running with hashes from previous blocks. In short, it took longer for the GO program test to complete than for the private network to validate a new block.

Therefore, it has been essential to implement asynchronous calls in the blockchain network, so that it has not need to wait for the API response and continues its normal operation. At the same time, and more importantly, a queuing system has had to be implemented in the GO program. This has been achieved using GO channels and GO routines. When the application launches, the channel is initialized and a GO routine is started to handle incoming requests. Then for each API call received, the hash information has been added to the channel queue, to be executed one at a time on a first-come, first-served basis.

Memory usage In addition to measuring the execution time of cryptographic functions, in order to assess whether an algorithm is suitable or not, it has been very interesting to quantify the memory usage during executions. For this purpose, four metrics have been evaluated:

- **Alloc**: Amount of memory allocated by the Go runtime for live objects, including all reachable objects in the heap, as well as some additional memory used by the garbage collector and other runtime structures.
- **Total alloc**: Cumulative amount of memory allocated since the program started. It covers all allocations, even those that have been freed by the garbage collector.
- **Sys**: Total memory obtained from the operating system, including both the Go heap and any memory allocated by the Go runtime for other purposes (such as stack space, memory-mapped files, and so on).
- **Num GC**: Number of garbage collection cycles that have occurred since the program started. Garbage collection is the process of reclaiming memory that is no longer in use by the program, and each cycle involves scanning the heap to identify and free unreachable objects.

5 Experimental results

The test results have been divided into three parts: the key and signature sizes, and the execution times and memory usage of the functions under test.

5.1 Key and signature sizes

Table 1 shows the comparison of private and public key and signature sizes for each type of cryptographic algorithm. It is worth mentioning that the size of the public key is only the size of the key itself, which in reality would have to be added 1 bit for the signature of the y-coordinate.

Comparing the key sizes of the various cryptographic schemes, there have been notable differences that reflect their security objectives and underlying algorithms. The best values have been obtained by ECDSA and 128-bit SPHINCS$^+$, which with the sum of the public and private key sizes achieve the same size. ECDSA has a 64-byte public key and a 32-byte private key, while SPHINCS$^+$ has a 32-byte public key and a 64-byte private key.

Table 1: Key and signature sizes of different algorithms

Algorithm	**Public key**	**Private key**	**Signature**
ECDSA	64	32	64
Dilithium2	1312	2528	2420
Dilithium3	1952	4000	3293
Dilithium5	2592	4864	4595
Falcon 1024	1793	2305	1231
SPHINCS$^+$ SHA256 128bit - Robust	32	64	17088
SPHINCS$^+$ SHA256 192bit - Robust	48	96	35664
SPHINCS$^+$ SHA256 256bit - Robust	64	128	49856

In any case, although ECDSA could be considered the best in this aspect, it should be noted that the set of all SPHINCS$^+$ schemes have values very close to ECDSA and therefore, values that could be competitive to it in this aspect. Among SPHINCS$^+$ versions, the higher the security level, the more bits are used and therefore, the key sizes increase. However, using the simplest version, the 128 bit one, it is possible to use the same key sizes as ECDSA.

In contrast, both Dilithium and Falcon have been excessively far from the ECDSA or SPHINCS$^+$ key size values. They should be used in scenarios where security is paramount and large key size is not an issue.

In case of the signatures, it can be seen that the result changes radically compared to the key sizes comparison. Although ECDSA has still been the best, SPHINCS$^+$, which for the keys was the second best option, becomes the worst with a huge difference with the rest.

In this case, the second best option has been Falcon 1024. However, the size have fallen far above of the ECDSA signature size, since the size of a Falcon 1024 signature is equivalent to just over 19 ECDSA signatures.

The size of the signatures generated using Dilithium has also turned out to be considerably larger, doubling the size of Falcon 1024 using Dilithium2, and more than tripling if Dilithium5 is employed.

In general, it can be concluded that there is no scheme that globally (adding the three sizes) comes close to ECDSA, since the second best is Falcon and exceeds it by 33 times. Especially analyzing the sizes of the signatures, it can be clearly stated that all post-quantum schemes are extremely far from the ECDSA values. However, SPHINCS$^+$, in the aspect of the keys obtains a very good result, being even better in the size of the public keys and not much worse in the private ones. In any case, it is clear that using these algorithms, the increase in the size of the keys and especially of the signatures is an inevitable consequence of the improvement in security they offer.

5.2 Execution times

One of the most important factors when choosing one cryptographic algorithm over another is the time required for the algorithm to execute the cryptographic

functions. Therefore, much emphasis has been placed on measuring and analyzing in a precise and detailed way the execution times required by each cryptographic scheme.

The final average results after more than 24 hours of running the blockchain network have been those summarized in the Table 2 and also visible in the historical data dashboards of the project.

Table 2: Average execution times by algorithms (ms)

Algorithm	**Key generation**	**Signature generation**	**Signature verification**
ECDSA	90.7	58.6	71.4
Dilithium2	80.5	149	17.3
Dilithium2-AES	102	124	16.5
Dilithium3	144	222	22.7
Dilithium3-AES	171	191	19.5
Dilithium5	201	251	33.6
Dilithium5-AES	254	207	29.6
Falcon 1024	90434	13162	166
SPHINCS$^+$ SHA256 128bit - Robust	5612	120583	7845
SPHINCS$^+$ SHA256 128bit - Simple	3376	72760	4569
SPHINCS$^+$ SHA256 192bit - Robust	8318	198359	11856
SPHINCS$^+$ SHA256 192bit - Simple	4977	118773	6777
SPHINCS$^+$ SHA256 256bit - Robust	26333	483158	14568
SPHINCS$^+$ SHA256 256bit - Simple	13074	240571	6957

Firstly, evaluating the differences in key generation times, it can be concluded that there have been two algorithms that clearly stand out above the rest: ECDSA and Dilithium2 (both in its normal version and in the AES version, which hardly varies). However, it should be noted that all versions of Dilithium have a relatively good execution time that did not deviate that much from the ECDSA times. In fact, the Dilithium2 version improves on the ECDSA result and achieves the shortest execution time of all the algorithms. However, both SPHINCS$^+$, in all its versions, and Falcon, differ exaggeratedly compared to the times of the rest, especially the latter.

Secondly, evaluating the signature generation times, there is no doubt that ECDSA is clearly above the rest of the algorithms, as it is slightly more than twice as fast as the second, Dilithium2-AES. Unlike the key generation, in this section Falcon 1024 (224 times the ECDSA time) has obtained a better result than SPHINCS$^+$ (2057 times the ECDSA time in the simplest mode), but they are still times that are not at all competitive compared to the times of ECDSA or even Dilithium.

However, at this point it must be taken into account the large difference that existed between the signature size of ECDSA and the signature sizes of the post-quantum algorithms, which greatly influences the time of signature generation.

Finally, as far as signature verification times are concerned, it is interesting to note that ECDSA did not obtain the best score on this point, being clearly outperformed by Dilithium in all its versions.

Dilithium2 has been more than four times faster than ECDSA, and Dilithium5, the version with the highest level of security, 2.1 times faster too. Falcon 1024, on the other hand, although it has been more than twice as slow as ECDSA, was not far behind. SPHINCS$^+$, in all its modes, has been the scheme that took an exorbitant amount of time compared to the rest, being 47.3 times slower than Falcon 1024 for example (128bits robust mode). However, although it is not a very considerable difference, a clear difference in times could be observed when using the robust or simple mode of SPHINCS$^+$, regardless of the number of bits used.

Summarizing the results, on the one hand, in key generation, Dilithium obtains a result very close to ECDSA, even improving its time in its Dilithium2 version. The rest of the algorithms need an extremely higher time and are not competitive at all. The same happens in the generation of signatures, although in this case Dilithium does not improve the time of ECDSA in any version, it is very close. The worst here is SPHINCS$^+$, while Falcon improves compared to key generation. The most remarkable thing happens in signature verification, where Dilithium is exceptionally better than the rest, including ECDSA, being between 2.1 and 4.1 times faster than ECDSA depending on the versions used.

Therefore, in the sum of all times, it can be concluded that Dilithium needs a similar amount of time as ECDSA, being practically the same time in the Dilithium2 version. Comparing it with the rest of the post-quantum schemes, it obtains an excellent result, being the only one that could compete in this aspect with the current cryptographic methods.

5.3 Memory usage

In addition to measuring the execution time of cryptographic functions, in order to assess whether an algorithm is suitable or not, it has been very interesting to quantify the memory usage during executions. Nevertheless, obtained results have not been as interesting and analyzable as those of the execution time, which has led to more extensive conclusions.

The values of total alloc, sys and num GC, have concluded to be very similar between all the executions of the different algorithms in each test. The metric of interest for the analysis has been the alloc, the amount of memory allocated by the Go runtime for live objects, which has led to detect greater variation among the different algorithms.

First, in the memory allocation levels of the key generation, a clear difference has been observed between ECDSA and Falcon compared to Dilithium and SPHINCS$^+$. Although ECDSA achieved the lowest result, it is less than 1% better than Falcon, which is a negligible difference. The difference to consider exists when compared to the other two post-quantum algorithms, which need about 55-60% more memory than the first two.

Next, the results of the same metric have been analyzed for the signature generation process, and it was observed that in this aspect, Dilithium improves and approaches the values of ECDSA and Falcon, which are still slightly better in that order. SPHINCS$^+$, on the other hand, still requires about 50% more memory compared to the other three algorithms.

Finally, signature verification follows the same pattern as signature generation, with all the algorithms being quite close to each other, except SPHINCS, which continues with much higher values. However, in this case, it can be seen that there is a notable difference between the different modes that has SPHINCS. The robust 192-bit mode gives the best result, while curiously, the simple mode of the same bits is the one that gives the worst result of all, surpassing it by just over 10%. The rest of the algorithms are in between, but except for the 128-bit one, in the other two, it is the simple mode which needs less allocated memory than the robust mode.

6 Conclusions and future work

One of the main conclusions was obtained during the research and study process. At present, quantum computers are still a long way from being able to break current encryption schemes easily and quickly. Current quantum computers are not necessarily large enough to break the schemes fast, nor are they anywhere near the size needed at the moment.

The test execution times have been where the most interesting information has been extracted, especially because a similar performance has been seen between ECDSA and Dilithium, and a huge difference has been observed with the other two analyzed schemes.

Analyzing the memory usage, it can be said that even though there is a difference between the schemes, it is not as big as in the case of the execution times. Falcon equals ECDSA in terms of results, followed by Dilithium, which needs more resources in key generation, but in the other two processes it is practically the same as ECDSA. However, SPHINCS$^+$ does require slightly more memory in all processes.

In summary, analyzing all the results, it can be clearly concluded that in terms of performance, Dilithium is currently the best post-quantum cryptographic scheme, being quite close to the performance levels of ECDSA, which has been the present scheme used as a comparison. In the case of the simplest version of Dilithium (Dilithium2), summing all the values of time and memory, it would only need 12% more execution time in the three processes evaluated and 20% more memory. These values, which although on a large scale could make a considerable difference, are values that are not too far off current levels. Therefore, the increase in security level that the integration of this algorithm could entail, could be totally understandable in exchange for the slight decrease in performance.

From the results presented and also during the development, new ideas, questions and improvements have arisen, which will be detailed below.

The main point for improvement would be to integrate the post-quantum cryptographic methods used in the tests into the blockchain network directly, i.e. to include them into the source code of the used Ethereum client, in this case Go-Ethereum. This way, more detailed tests could be performed, with real transactions, enabling the analysis of more factors such as network load for example. It would be possible to analyze the behavior of the methods in a real network, including latency, node distribution, volume of transactions...

Secondly, while current tests only include post-quantum digital signature algorithms selected by NIST, many other post-quantum algorithms could be tested in the future.

Finally, current tests were performed on a local computer with its own computational limitations and concurrent processes, which might affect results. Moving the system to an external server dedicated to testing, free from other tasks, would provide more accurate performance measurements. Additionally, creating a network of multiple systems to run the same tests and share a common database would allow for calculating average values across different systems, offering a broader perspective on performance results.

Acknowledgements

This work has been partially supported by project PID2022-139268OB-I00, financed by MCIN/AEI /10.13039/501100011033 / FEDER, UE and project TED2021-129830B-I00, financed by MCIN/AEI /10.13039/501100011033/Next-GenerationEU/PRTR.

References

1. Arute, F., Arya, K., Babbush, R., et al.: Quantum supremacy using a programmable superconducting processor. Nature **574**(7779), 505–510 (10 2019). https://doi.org/10.1038/s41586-019-1666-5
2. Bhatia, V., Ramkumar, K.: An efficient quantum computing technique for cracking rsa using shor's algorithm. In: 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA). pp. 89–94 (2020). https://doi.org/10.1109/ICCCA49541.2020.9250806
3. Brooks, M.: Quantum computing is taking on its biggest challenge: noise (01 2024), https://www.technologyreview.com/2024/01/04/1084783/quantum-computing-noise-google-ibm-microsoft/
4. Buser, M., Dowsley, R., Esgin, M., Gritti, C., Kermanshahi, S.K., Kuchta, V., Legrow, J., Liu, J., Phan, R., Sakzad, A., Steinfeld, R., Yu, J.: A survey on exotic signatures for post-quantum blockchain: Challenges and research directions. ACM Computing Surveys **55**(12) (3 2023). https://doi.org/10.1145/3572771
5. Chen, J., Gan, W., Hu, M., Chen, C.M.: On the construction of a post-quantum blockchain for smart city. Journal of Information Security and Applications **58**, 102780 (2021). https://doi.org/10.1016/j.jisa.2021.102780
6. Cloudflare: Circl: Cloudflare interoperable reusable cryptographic library (04 2024), https://pkg.go.dev/github.com/cloudflare/circl/sign/dilithium
7. CRYSTALS Team: Cryptographic suite for algebraic lattices (crystals) (08 2023), https://pq-crystals.org/

8. Daugaard, K.: Sphincsplus-golang (12 2023), https://github.com/kasperdi/SPHINCSPLUS-golang
9. Fernández-Caramès, T.M., Fraga-Lamas, P.: Towards post-quantum blockchain: A review on blockchain cryptography resistant to quantum computing attacks. IEEE Access **8**, 21091–21116 (2020). https://doi.org/10.1109/ACCESS.2020.2968985
10. Fouque, P., Hoffstein, J., Kirchner, P., Lyubashevsky, V., et al.: Falcon (11 2017), https://falcon-sign.info/
11. Gao, Y.L., Chen, X.B., Chen, Y.L., Sun, Y., Niu, X.X., Yang, Y.X.: A secure cryptocurrency scheme based on post-quantum blockchain. IEEE Access **6**, 27205–27213 (2018). https://doi.org/10.1109/ACCESS.2018.2827203
12. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions (2007), https://eprint.iacr.org/2007/432
13. Gidney, C., Ekerå, M.: How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. Quantum **5**, 433 (4 2021). https://doi.org/10.22331/q-2021-04-15-433
14. IBM: Ibm's roadmap for scaling quantum technology. IBM (12 2023), https://www.ibm.com/quantum/blog/ibm-quantum-roadmap
15. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ecdsa). International Journal of Information Security **1**(1), 36–63 (08 2001). https://doi.org/10.1007/s102070100002
16. Li, C.Y., Chen, X.B., Chen, Y.L., Hou, Y.Y., Li, J.: A new lattice-based signature scheme in post-quantum blockchain network. IEEE Access **7**, 2026–2033 (2019). https://doi.org/10.1109/ACCESS.2018.2886554
17. National Institute of Standards and Technology (NIST): Post-quantum cryptography (04 2024), https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022
18. Pati, C.: Search using grover's algorithm. National Institute of Technology Rourkela (11 2023). https://doi.org/10.13140/RG.2.2.25842.07369
19. Pornin, T.: Falcon source files (reference implementation) (11 2021), https://falcon-sign.info/impl/falcon.h.html
20. SPHINCS+ Team: Sphincs+ (08 2023), https://sphincs.org/
21. Yang, Z., Alfauri, H., Farkiani, B., Jain, R., Di Pietro, R., Erbad, A.: A survey and comparison of post-quantum and quantum blockchains. IEEE Communications Surveys & Tutorials **26**(2), 967–1002 (2024). https://doi.org/10.1109/COMST.2023.3325761
22. Yi, H.: Secure social internet of things based on post-quantum blockchain. IEEE Transactions on Network Science and Engineering **9**(3), 950–957 (2022). https://doi.org/10.1109/TNSE.2021.3095192