Privacy-preserving decision tree inference under CKKS: A precision impact analysis

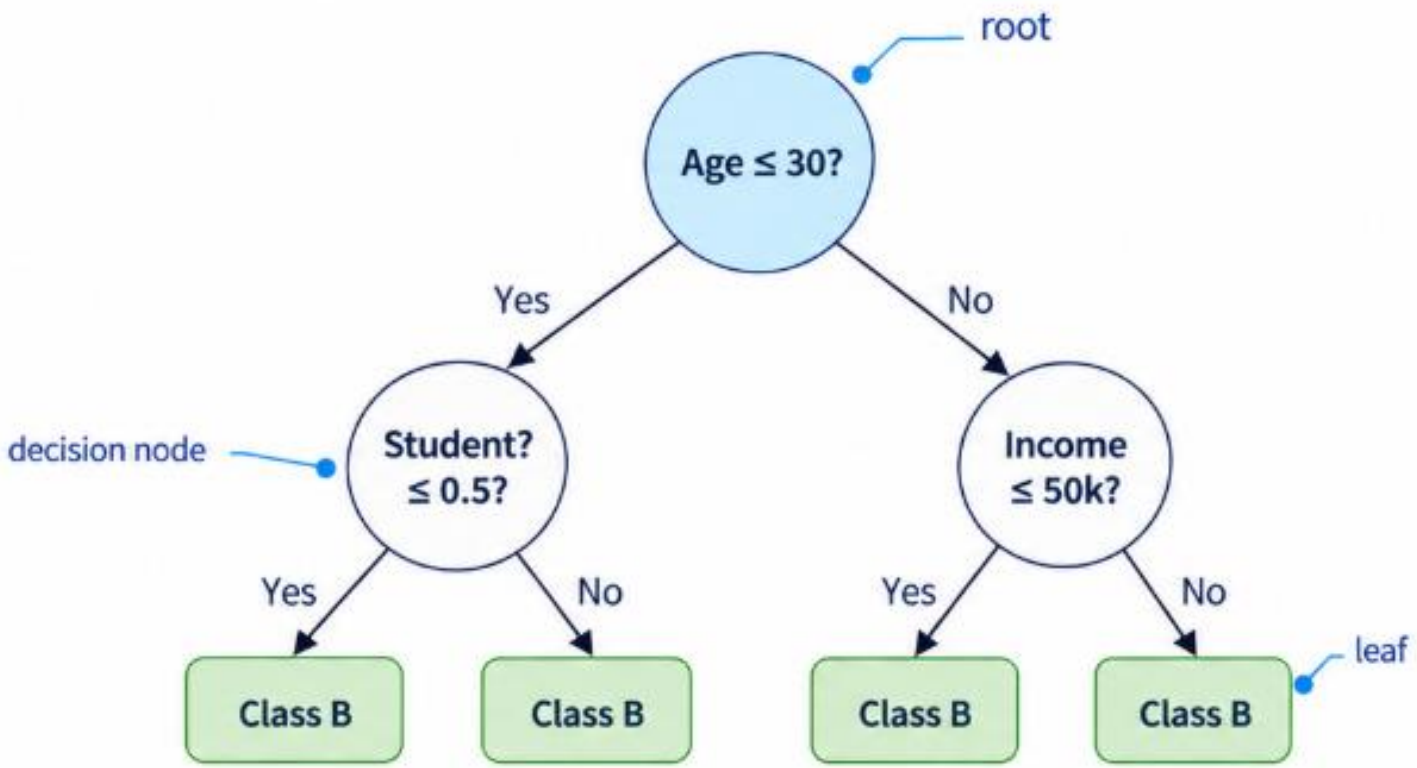
Madické Diadji Mbodj, Mawloud Omar, Reda Yaich, Anis Bkakria, and Siham Bouchelaghem

Outline

- Recall:
  - Decision tree,
  - Homomorphic encryption: CKKS
  - Global approach (SoTA)
- Contributions
  - Soft-Adaptive degree assignment,
  - Precision-impact analysis,
  - End-to-end proof of concept
- Conclusions and perspectives

Decision tree : quick reminder

Machine Learning Basics



1. A supervised learning model
2. Each internal node tests a feature with a threshold
3. Each branch represents an outcome
4. Each leaf gives a prediction

Why use decision trees?

Interpretable

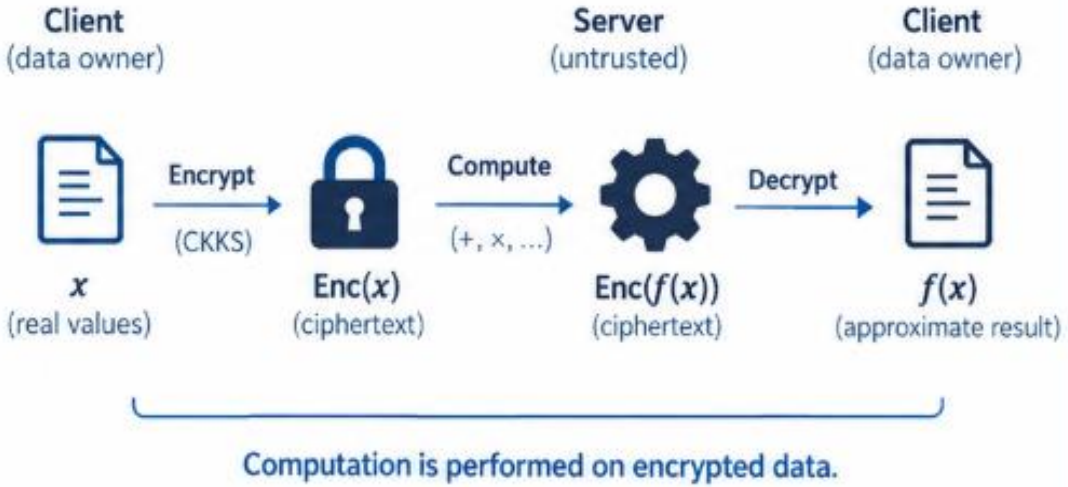
Simple

Non-linear decisions

Homomorphic encryption: CKKS

CKKS
(Cheon–Kim–Kim–Song)

- Homomorphic encryption scheme for approximate arithmetic on real numbers.
- Enables computation directly on encrypted data without decrypting.
- Well-suited for machine learning and inference tasks.



Key properties

- **Homomorphic:** compute on ciphertexts (no decryption).
- Approximate arithmetic on real numbers.
- Suitable for privacy-preserving machine learning.
- Trade-off between precision, performance and security.

Main CKKS parameters

Polynomial modulus degree N
Controls capacity / number of slots and security level.

Coefficient modulus Q
Determines the number of levels and the precision budget.

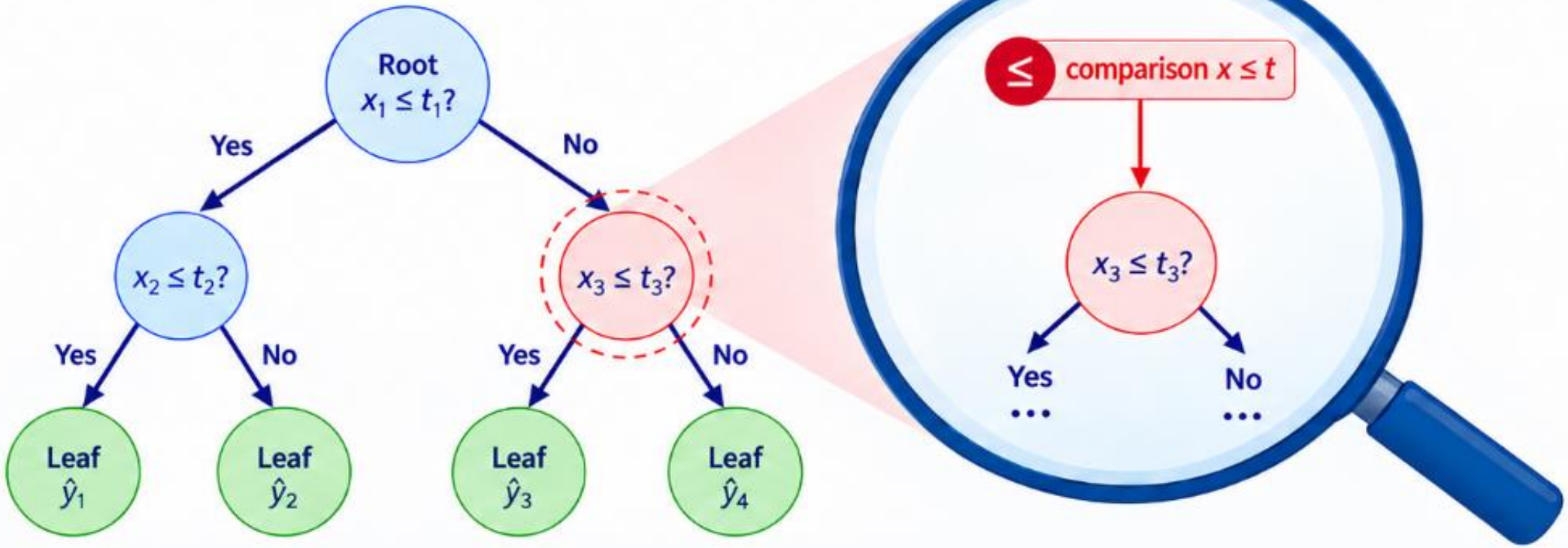
Scale Δ
Sets the numerical precision.

Multiplicative depth L
Maximum number of successive multiplications.

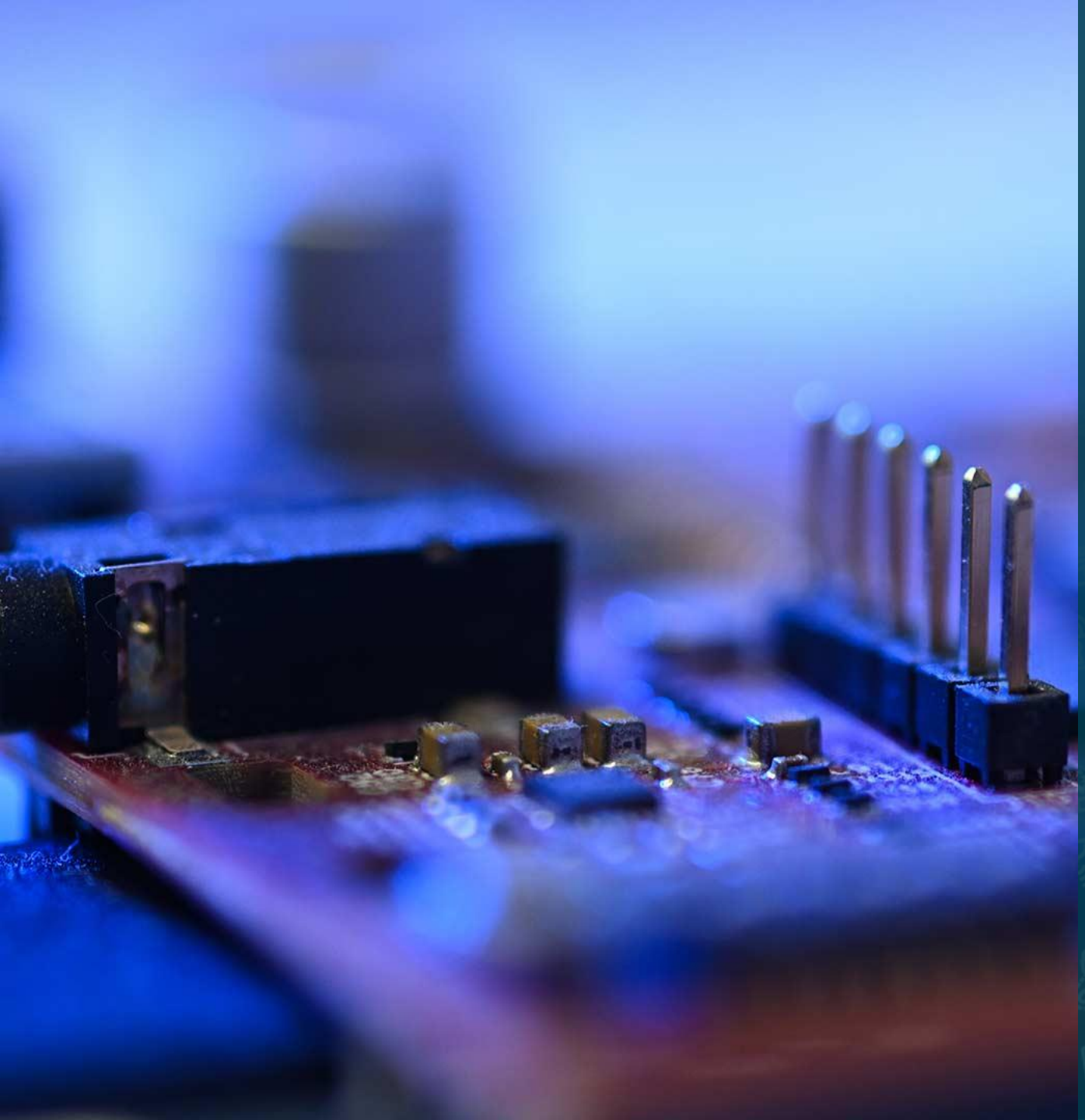
DECISION TREES ON ENCRYPTED DATA

CKKS bottleneck for decision trees

The bottleneck appears at node comparisons and path selection



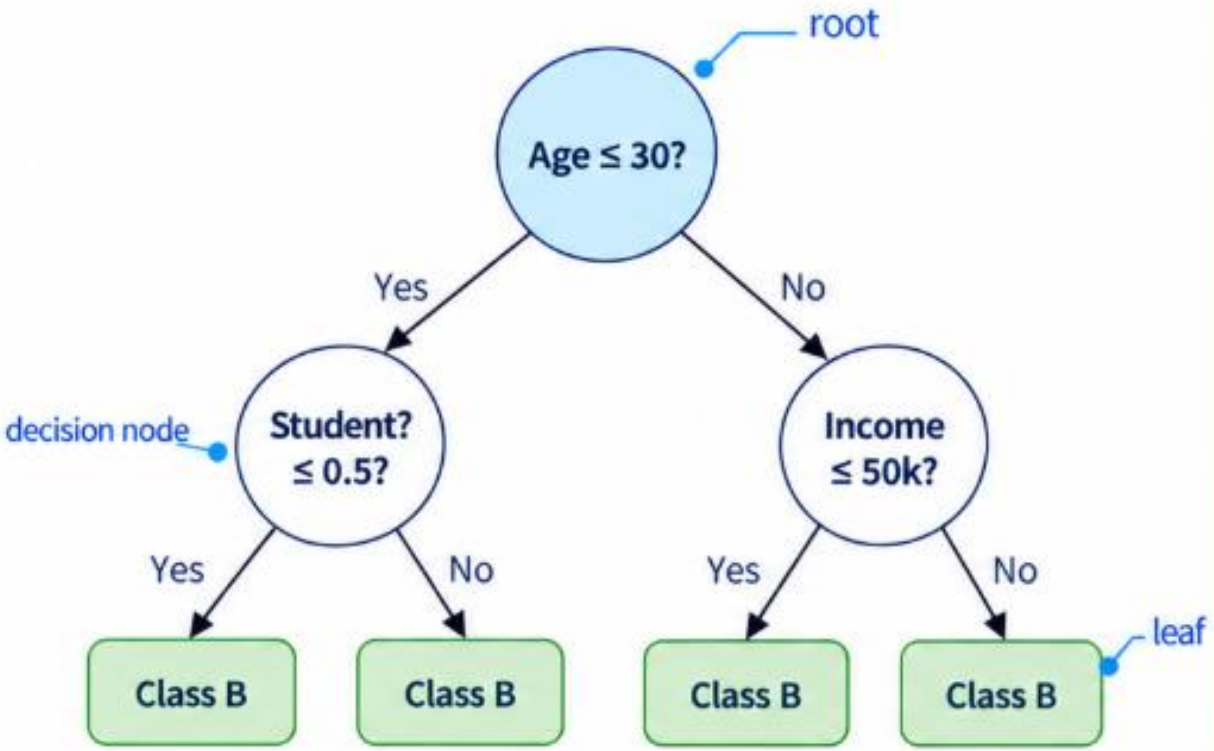
Main idea: tree inference must be rewritten as arithmetic operations, and node comparisons, branching, and noise make decision trees challenging under CKKS.



Global approach

State of the art

Comparators and tree traversals



Comparator families

- 0 1 0 1 / 1 0 1 0 — Bitwise comparison
- Direct arithmetic comparison
- Used — Approximate or continuous comparison
- Inclusion-based and combinatorial encodings

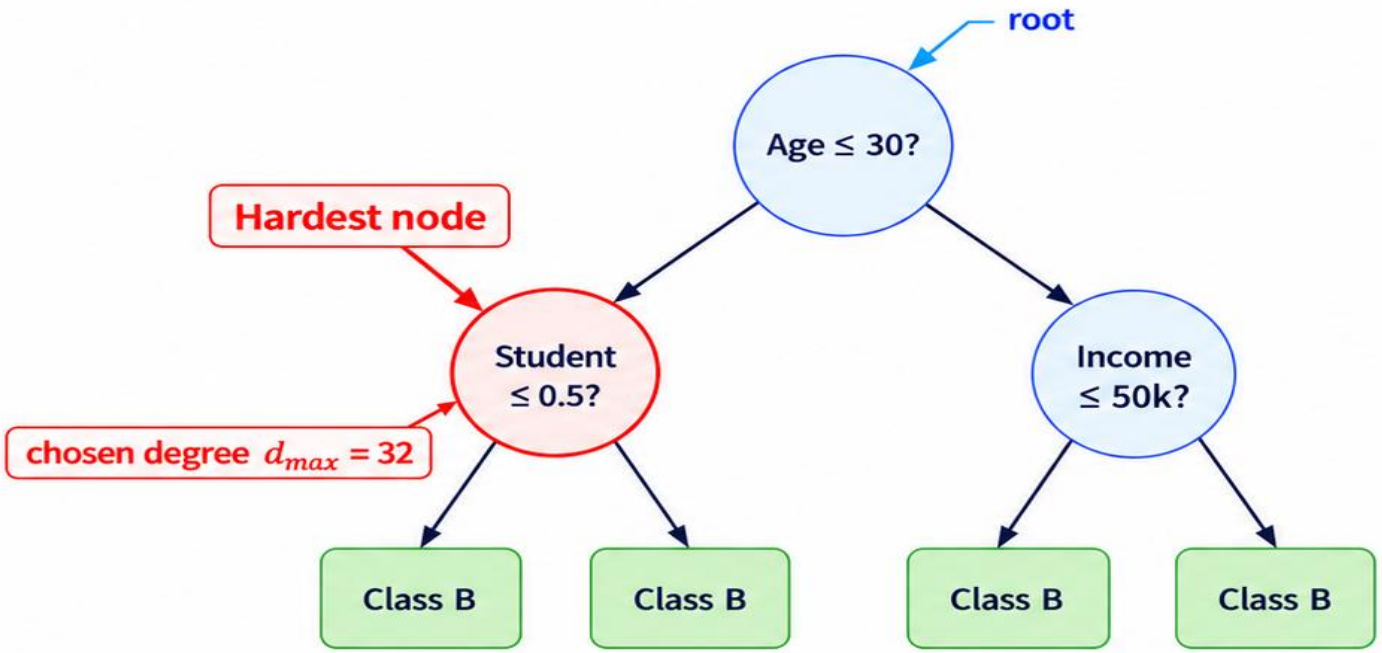
Tree traversal families

- Polynomial-evaluation traversal
- Used — Path-indicator-vector traversal (SumPath)
- Matrix-based traversal

Our method uses an approximate/continuous comparator together with Path-Indicator-Vector traversal (SumPath).

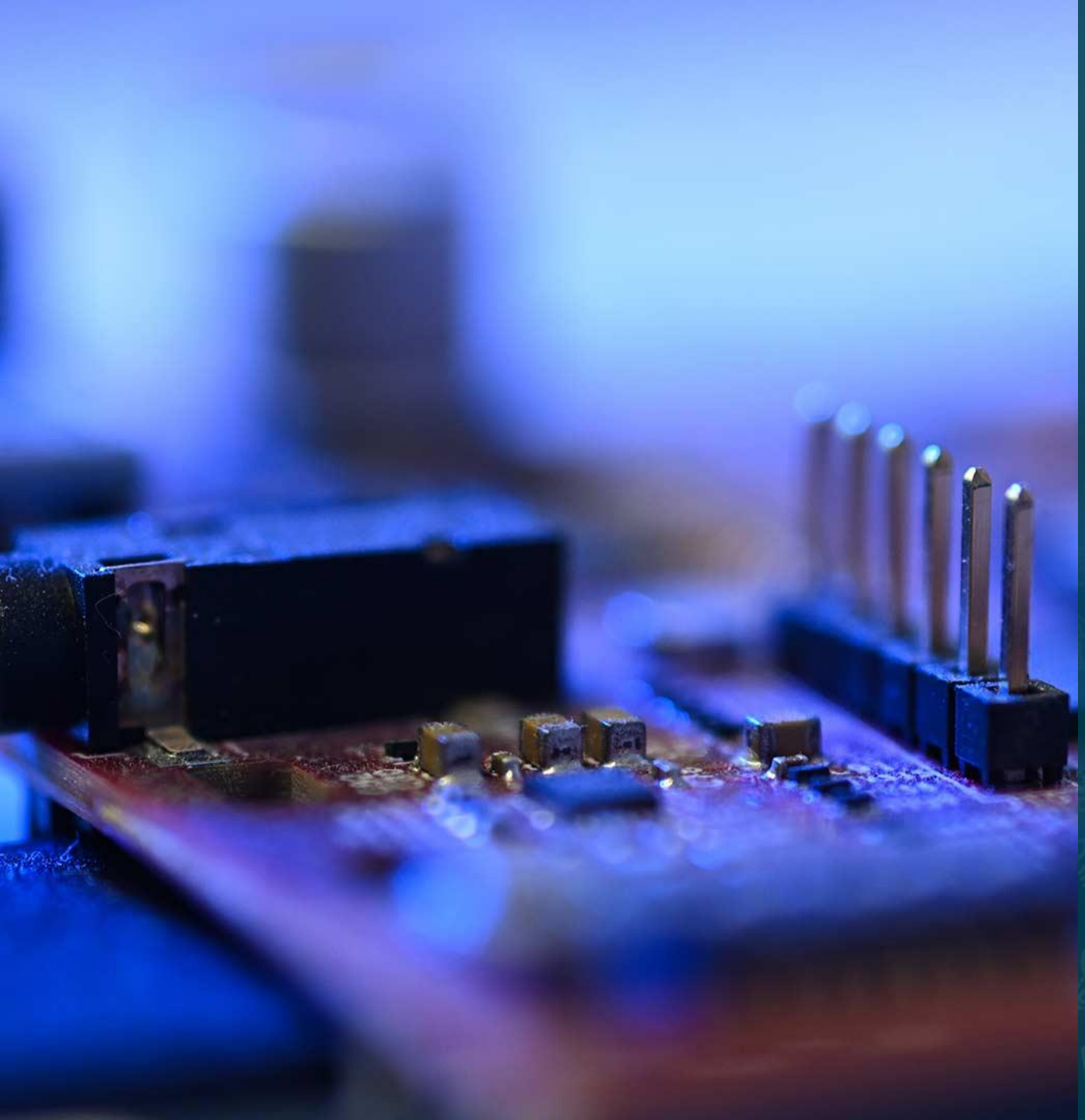
Global polynomial approximation in prior work

The same polynomial comparator is used for every internal node



1. Find the hardest node (feature closest to its threshold).
2. Choose a maximal polynomial degree d_{max} for good precision.
3. Use the same polynomial approximation for all internal nodes.

Main idea: select d_{max} from the hardest node, then reuse the same degree for all internal nodes.



Contributions

- Soft-Adaptive degree Polynomial
- Precision impact of analysis
- Proof of concept

Soft Step function

Comparators Soft Step polynomial comparison (Akavia et al.[1]):

$$\phi_d = \arg\min_{p \in \mathcal{P}_d} \int_{-2}^{2} \left(I_0(t) - p(t)\right)^2 w_\delta(t)\, dt \quad .$$

$w_\delta(t) = \mathbf{1}[|t| \geq \delta]$, the neglected window.

$I_0(t) = \mathbf{1}[t \geq 0]$, the Heaviside function

HBDT-SumPath Traversal (Shin et al.[2]):

$$S_{\text{mis}}(L_k) = \sum_{j=1}^{D_k} \left[b_{k,j} \left(1 - I_{i_j}\right) + \left(1 - b_{k,j}\right) I_{i_j} \right].$$

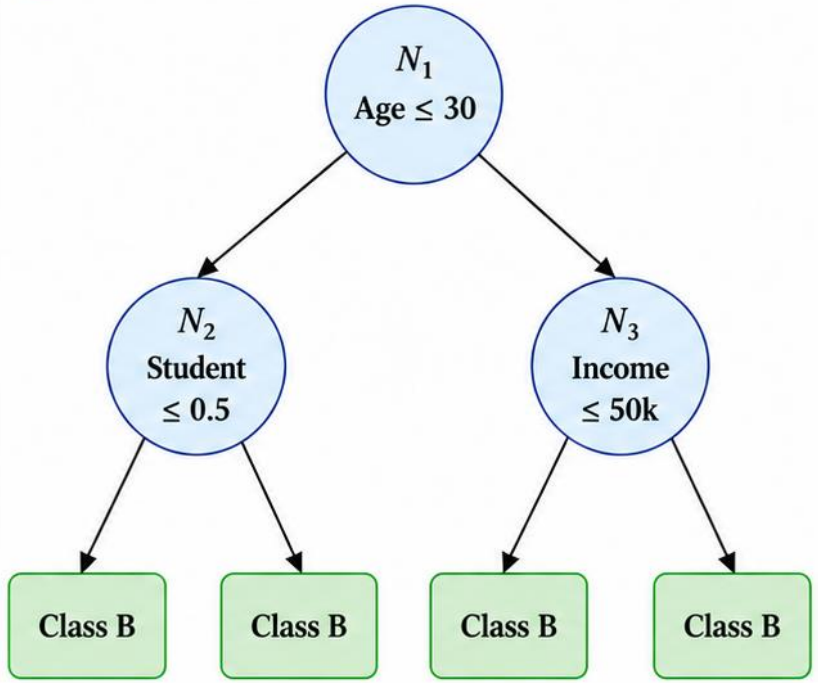
The selected leaf is : $arg\ min_k S_{mis}(L_k) \approx 0.$

Soft- adaptive degree assignment

Preparing the Soft-Adaptive Pipeline

Step 1 — From the trained tree to parameter extraction

Trained decision tree (example)



Extraction of internal nodes

Extracted parameters of internal nodes

Node ID	Feature index	Threshold	Path depth
N_1	f_1 (Age)	30	1
N_2	f_2 (Student)	0.5	2
N_3	f_3 (Income)	50k	2

Step 1: Trained tree → Explicit parameters per internal node → Ready for the next stage (Soft-Adaptive)

Soft- adaptive degree assignment

Préparing the Soft-Adaptive Pipeline

Step 2 — Compute the minimum feature gap and normalize it

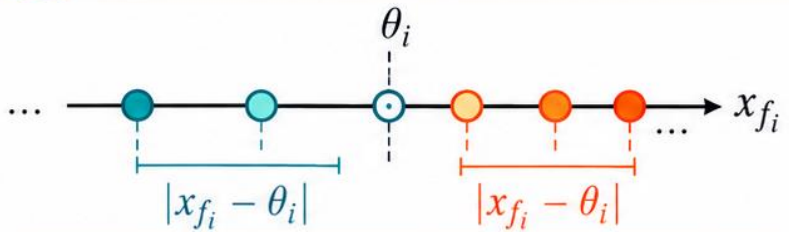
Selected internal node

$$N_i$$
$$f_i \le \theta_i$$

Feature index: f_i

Threshold: θ_i

Compute the gap



$$g_i = \min_{x \in X_{calib,i}} |x_{f_i} - \theta_i| \quad \text{(minimum gap)}$$

$$\nu_i = \max_{x \in X_{calib,i}} |x_{f_i} - \theta_i| \quad \text{(scale)}$$

$$\bar{g}_i = \frac{g_i}{\max(\nu_i, \varepsilon_0)} \quad \text{(normalized gap)}$$

Use $\bar{g}_i$ when raw gaps are not directly comparable across nodes.

Interpretation

Large $\bar{g}_i$
→ easy node
→ lower degree

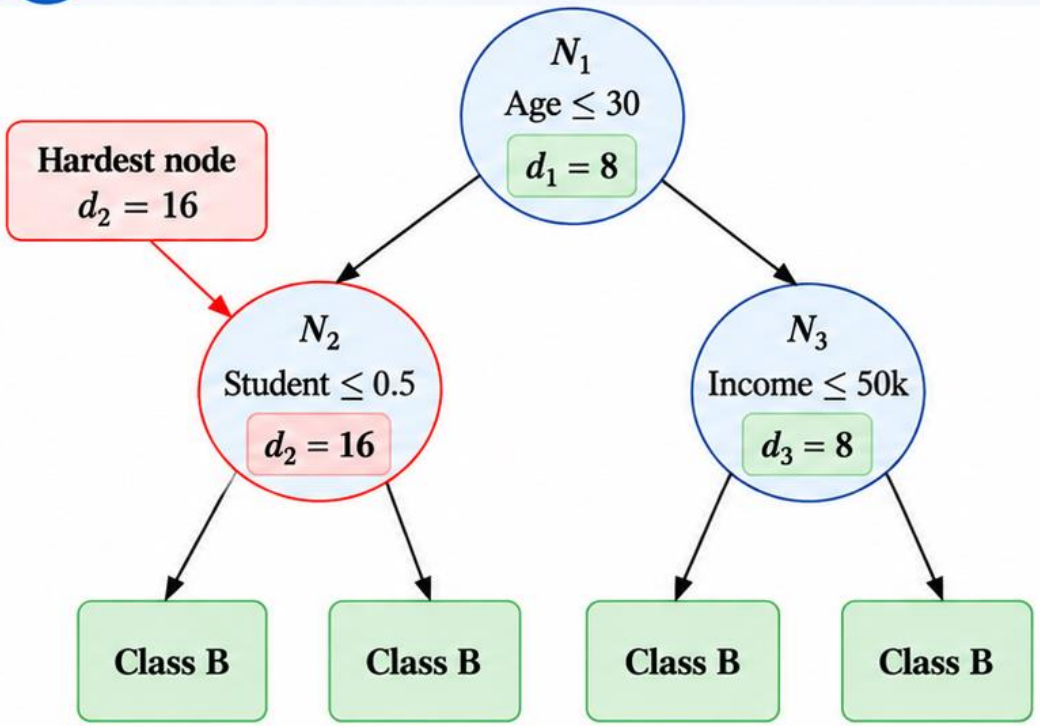
Small $\bar{g}_i$
→ difficult node
→ higher degree

For each node N_i → Compute g_i → Normalize into $\bar{g}_i$ → Choose the polynomial degree

Soft- adaptive degree assignment

Step 3 — Adaptive Polynomial Degree Assignment

Trained decision tree (with adaptive degrees)



Node difficulty and assigned degree

Node	Feature	Threshold θ_i	Min. gap g_i	Assigned degree d_i
N_1	Age	30	0.12	8
N_2	Student	0.5	**0.01**	**16**
N_3	Income	50k	0.08	8

Degree assignment rule (example)

$$d_i = \begin{cases} 4 & \text{if } \bar{g}_i \geq 0.1 \\ 8 & \text{if } 0.02 \leq \bar{g}_i < 0.1 \\ 16 & \text{if } 0.005 \leq \bar{g}_i < 0.02 \\ 32 & \text{if } \bar{g}_i < 0.005 \end{cases}$$

If g_i is outside the expected range, we use the normalised value:

$$\bar{g}_i = \frac{g_i}{\max(\nu_i, \epsilon_0)}$$

Key takeaway: Each node receives a polynomial degree based on the (normalised) minimum feature gap. The hardest node (N_2) requires the highest degree ($d_2 = 16$).

Soft- adaptive degree assignment

Step 4 — Building the SoftStep Comparator

1 Encrypted comparison margin for node N_i

N_i

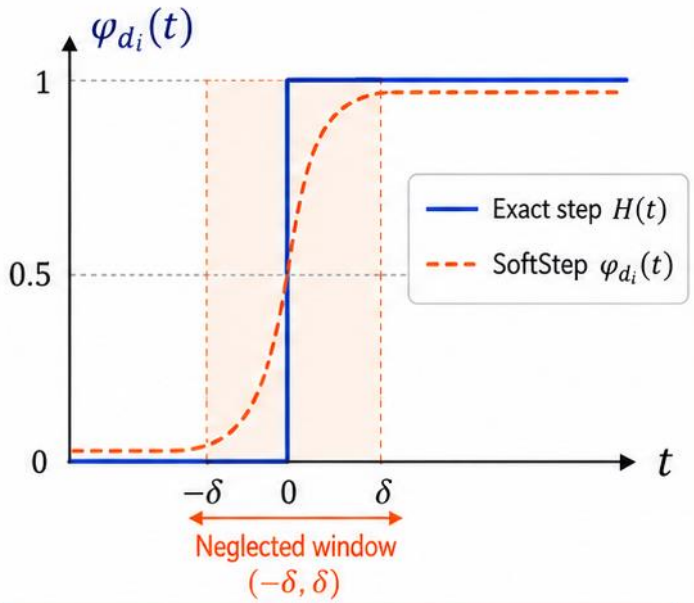
$x_{f_i} \le \theta_i$

$$t_i(x) = \theta_i - x_{f_i}$$

$t_i(x) > 0$: left side
$t_i(x) < 0$: right side
$t_i(x) = 0$: decision boundary

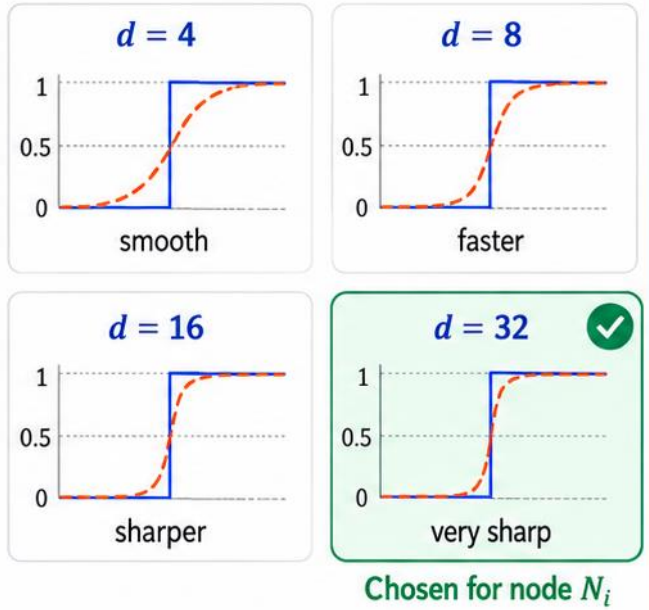
$t_i(x)$ remains encrypted.

2 Exact step vs. SoftStep approximation



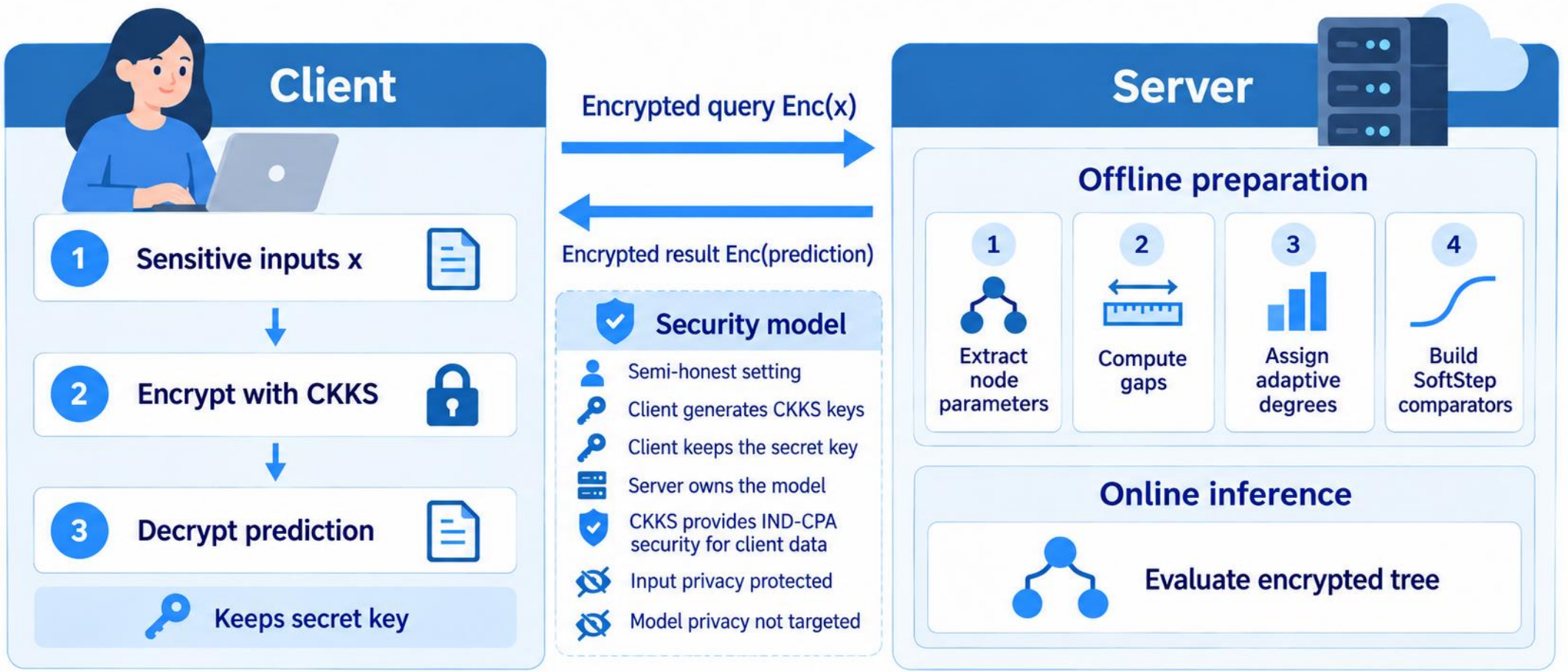
Higher degree → sharper transition.

3 SoftStep family $\varphi_d(t)$

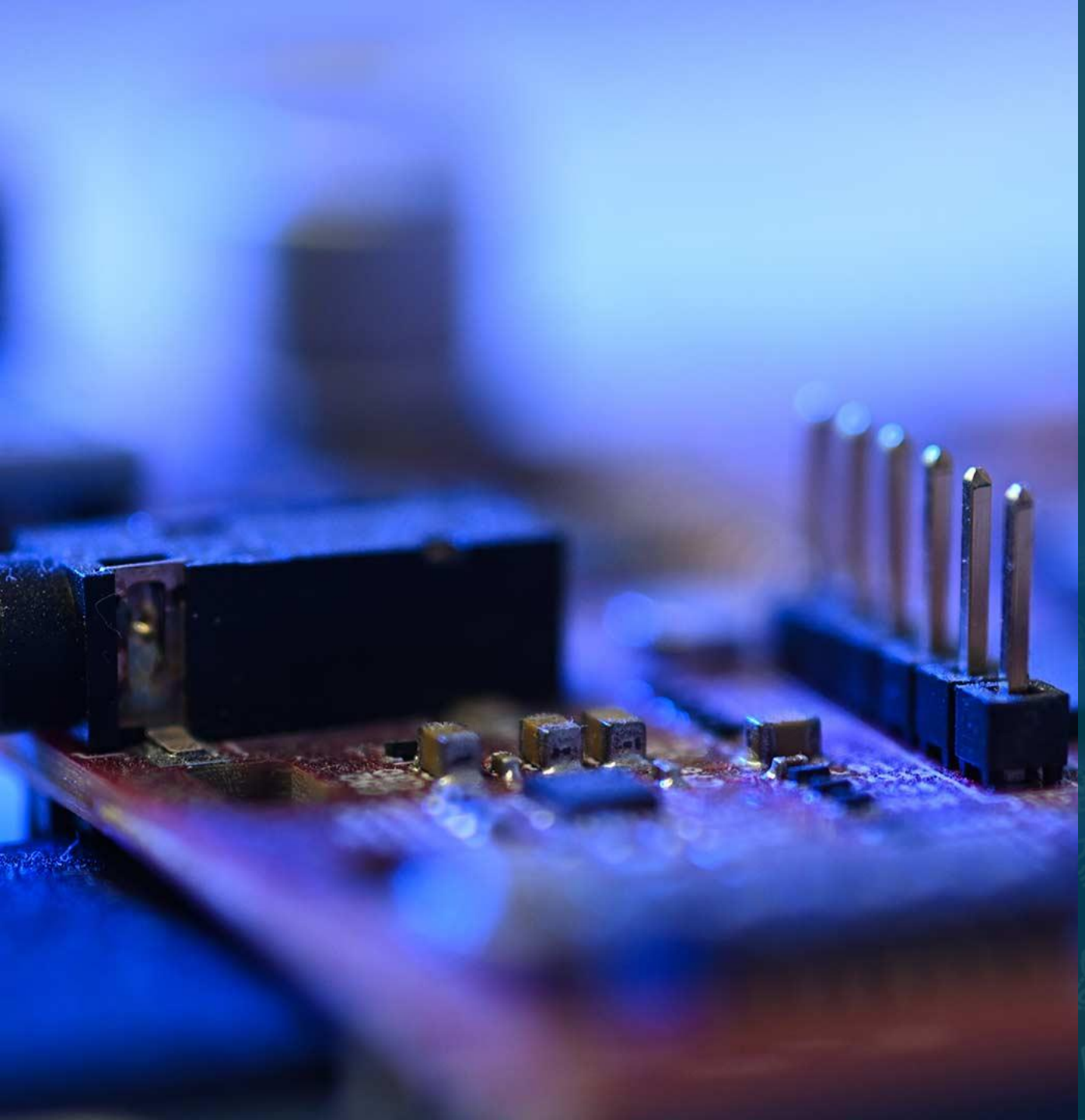


Encrypted margin $t_i(x)$ → SoftStep approximation $\varphi_{d_i}(t)$ → Polynomial comparator ready for CKKS

Output of the preparation phase: **one polynomial comparator per node.**



The server prepares the adaptive-degree pipeline once, then answers encrypted queries.



Precision-impact of analysis

Precision-impact of analysis

Let A_{Hard}, A_{soft}, and A_{HE} denote the accuracies of hard cleartext, Soft-Adaptive cleartext, and Soft-Adaptive HE inference. We decompose the total loss as:

$$(1)\ \Delta_{approx} = \mathrm{A}_{hard} - \mathrm{A}_{soft}$$

$$(2)\ \Delta_{CKKS} = \mathrm{A}_{soft} - \mathrm{A}_{HE}$$

$$(3)\ \Delta_{total} = \Delta_{hard} - \Delta_{HE} = \Delta_{approx} + \Delta_{CKKS}$$

- The approximation error outside the neglected window is

$$\varepsilon_{\text{poly}}(d,\delta) = \max_{|t|\geq\delta} |\phi_d(t) - I_0(t)| .$$

- The level-wise error of CKKS is :

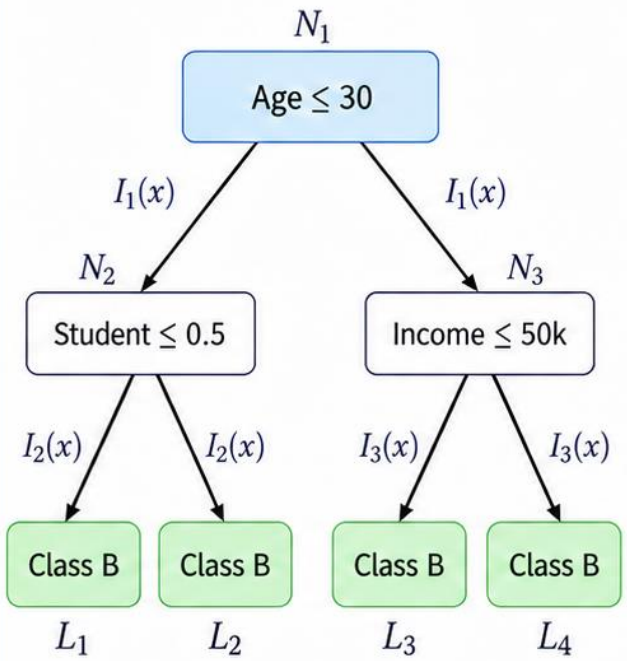
$$\varepsilon_\ell \approx \ell\sqrt{N}\,\Delta^{-1},$$

where ℓ is the number of consumed multiplicative levels, N is the polynomial modulus degree and the scale $\Delta = 2^s$.

.

How SumPath Works and Its Error Bound

How SumPath works



1. Each internal node outputs a soft value $I_i(x) \in [0, 1]$

2. For a leaf, SumPath adds the node contributions along its path

$$\hat{S}(L_k) = \sum_{j=1}^{D_k} |I_{ij}(x) - b_{k,j}|$$

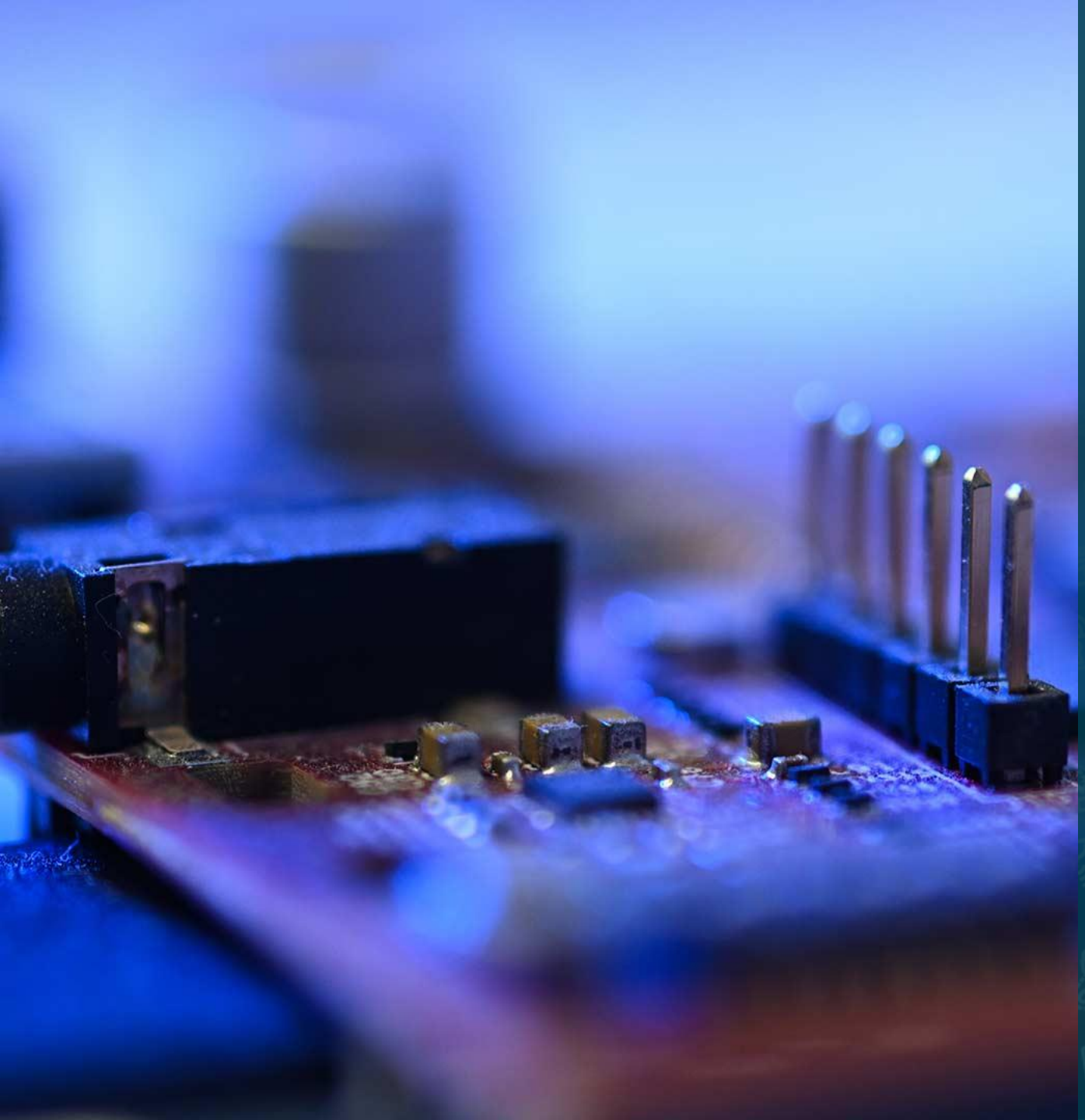
3. Predict the leaf with the smallest score

$$\hat{y} = \arg\min_{L_k} \hat{S}(L_k)$$

Theorem (SumPath error bound)

$$\left|\hat{S}^{HE}(L_k) - S^{bool}(L_k)\right| \leq \sum_{j=1}^{D} \varepsilon_{ij}^{total} \leq D \max_i \varepsilon_i^{total}$$

- Errors accumulate additively along the path.
- The bound grows linearly with the path depth D.



Experimental evaluation

Setup

Implementation & Environment

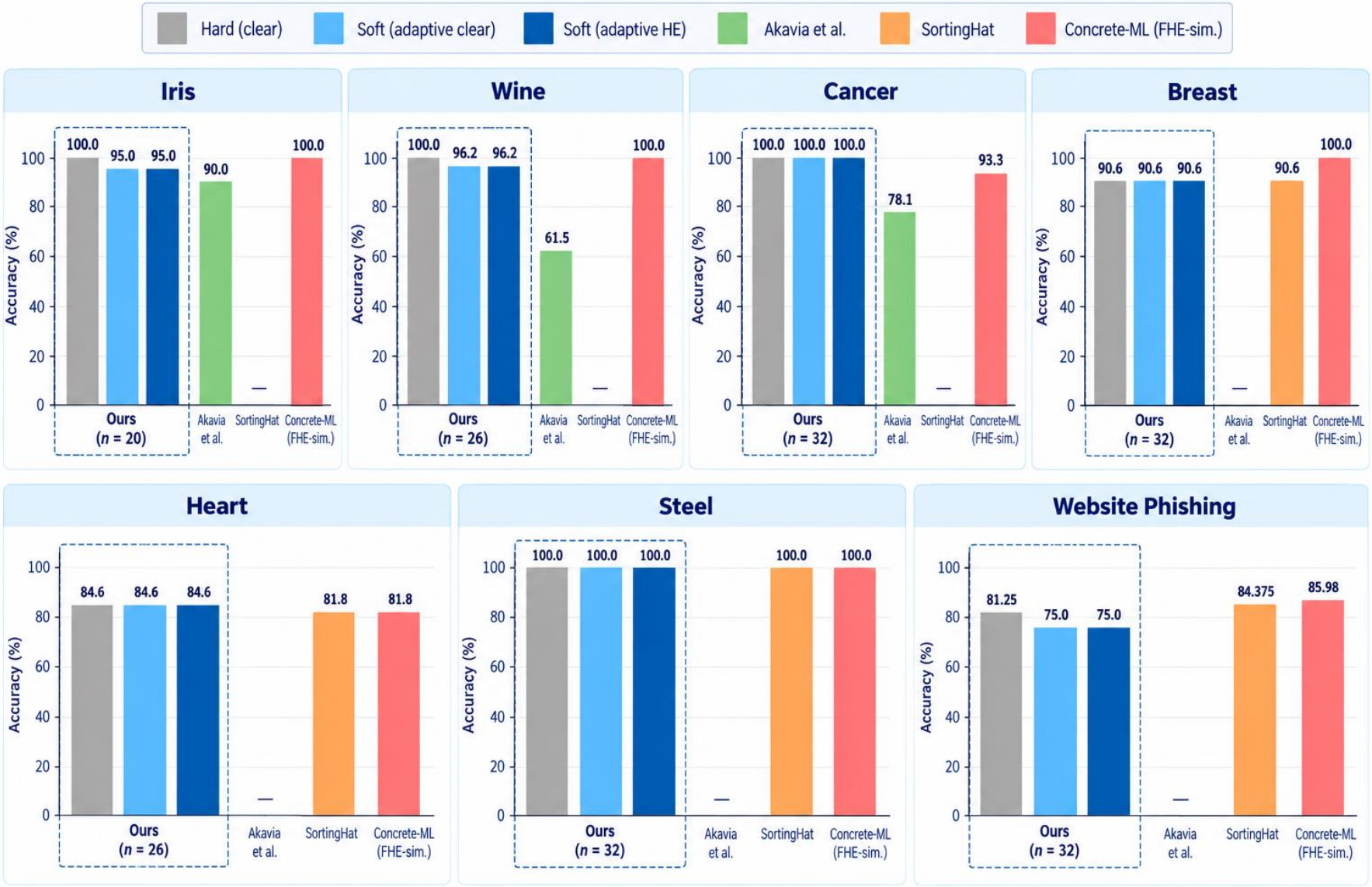
- Microsoft SEAL v4.1.2
- 128-bit security
- Ubuntu 22.04
- CPU-only execution
- Podman container
- C++17 / CMake
- Python / scikit-learn
- Vertical SIMD packing
- Max batch size: 32

CKKS Profiles

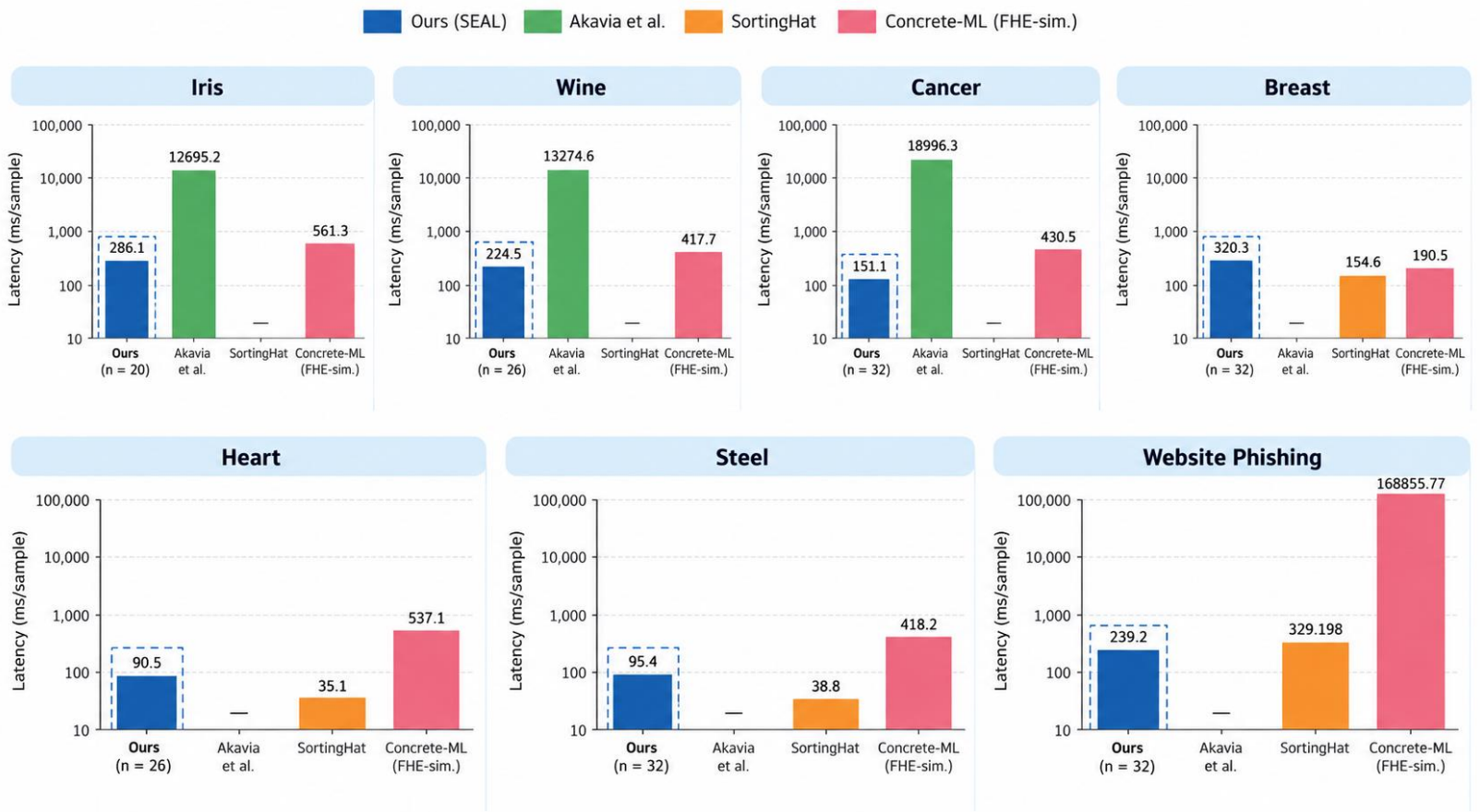
Small	Medium	Large
Feature dimension $dim \leq 4$	Feature dimension $5 \leq dim \leq 16$	Feature dimension $dim > 16$
$n_{\max} = 4$	$n_{\max} = 2$	$n_{\max} = 2$
Max adaptive degree $= 32$	Max adaptive degree $= 16$	Max adaptive degree $= 8$

Three runtime profiles keep encrypted inference feasible as feature dimension increases.

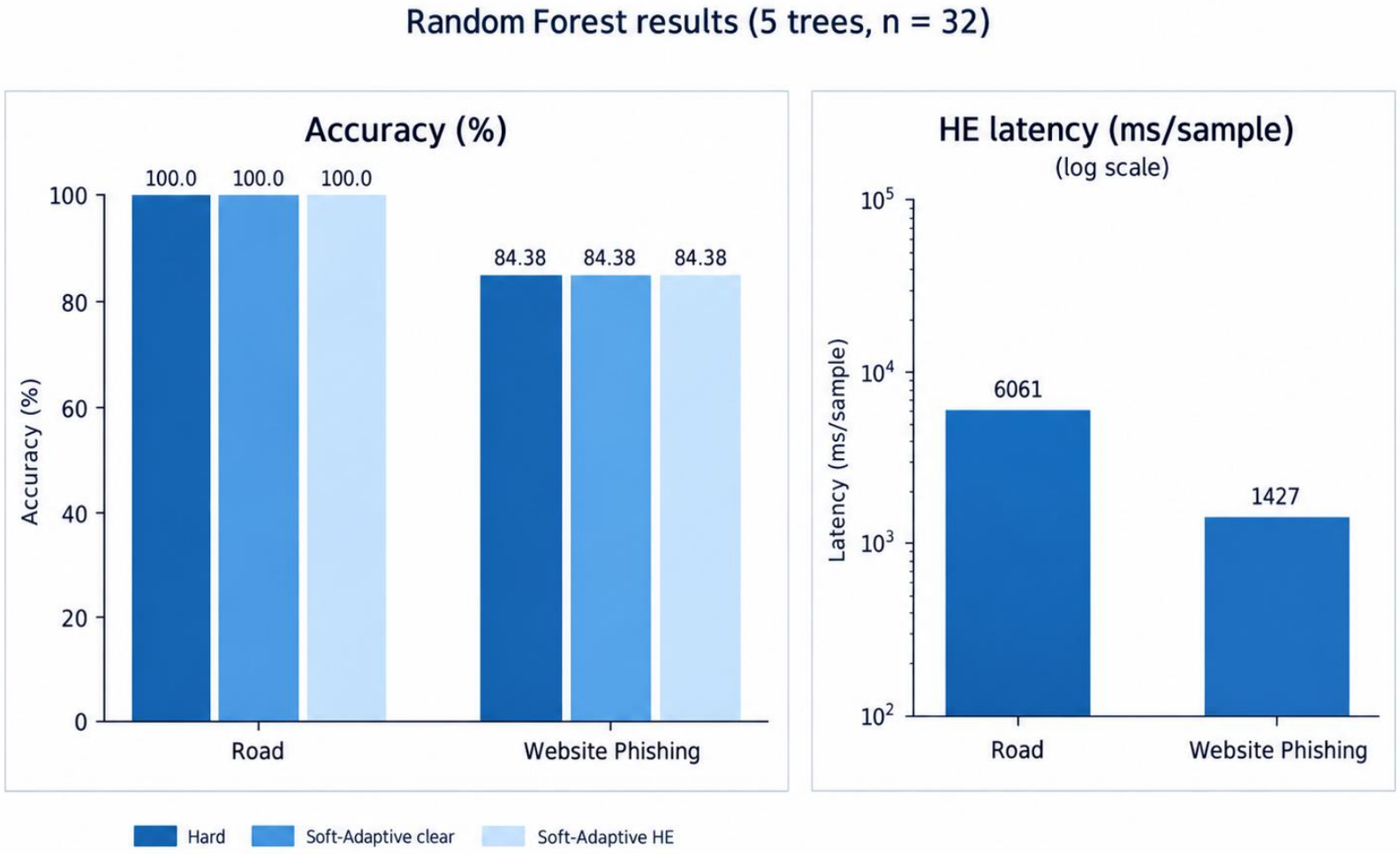
Results: accuracy for simple trees

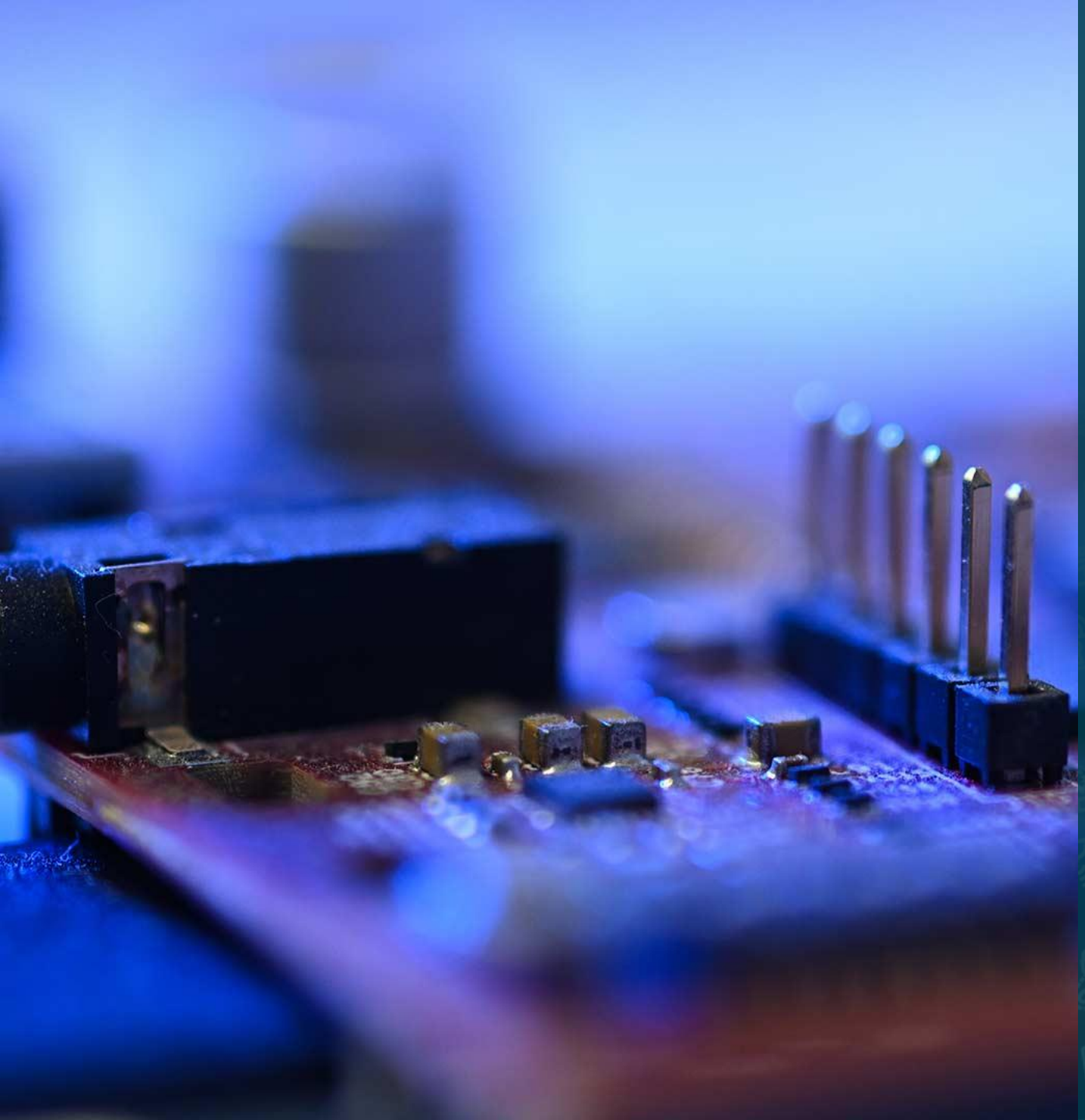


Results: latency for simple trees



Results : Random Forest





Conclusions and perspectives

Conclusions

Key contributions

Node-wise adaptive degree assignment

A polynomial degree is chosen locally for each node.

Precision-impact analysis

Separates approximation loss from CKKS-induced loss.

End-to-end CKKS inference

SoftStep comparators combined with SumPath traversal.

Accuracy insight

The main accuracy loss appears in the cleartext polynomial approximation, especially on Iris, Wine, and Website Phishing.

Soft-Adaptive HE matches Soft-Adaptive clear in the retained experiments.

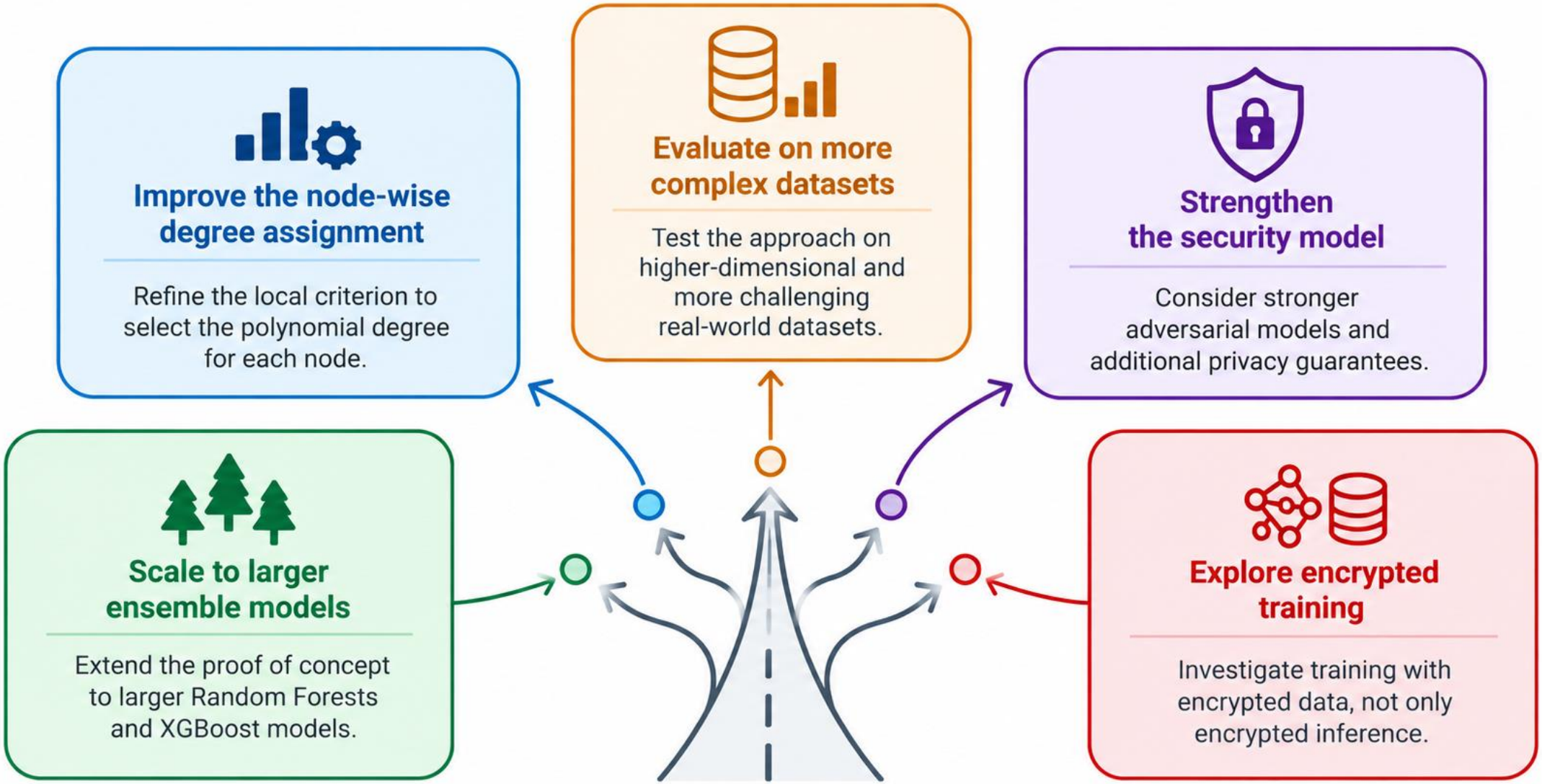
Latency insight

Latency is reasonable for single decision trees.

Latency becomes much higher for Random Forests.

Adaptive degrees help **preserve precision** while avoiding unnecessary complexity; scalability remains the **main challenge** for tree ensembles.

Perspectives



References

1. Akavia, A., Leibovich, N., Resheff, Y.S., Ron, R., Shahar, M., Vald, M.: Privacy-preserving decision trees training and prediction. ACM Transactions on Privacyand Security 25(3), 1–30 (2022)
2. Shin, H., Choi, J., Lee, D., Kim, K., Lee, Y.: Fully homomorphic training andinference on binary decision tree and random forest
3. Shin, H., Choi, J., Lee, D., Kim, K., Lee, Y.: Fully homomorphic training andinference on binary decision tree and random forest. In: European Symposium onResearch in Computer Security, pp. 217–237. Springer (2024)
4. Cong, K., Das, D., Park, J., Pereira, H.V.L.: SortingHat: Efficient private decisiontree evaluation via homomorphic encryption and transciphering. In: Proceedings ofthe 2022 ACM SIGSAC Conference on Computer and Communications Security,pp. 1647–1660. ACM (2022).
5. Cong, K., Kang, J., Nicolas, G., Park, J.: Faster private decision tree evaluationfor batched input from homomorphic encryption. In: International Conference on Security and Cryptography for Networks. Springer (2024)