Hindawi
Mobile Information Systems
Volume 2018, Article ID 2159082, 16 pages
https://doi.org/10.1155/2018/2159082

Review Article

Cryptocurrency Networks: A New P2P Paradigm

Sergi Delgado-Segura, Cristina Pérez-Solà, Jordi Herrera-Joancomartí, Guillermo Navarro-Arribas, and Joan Borrell

Department d'Enginyeria de la Informació i les Comunicacions, Universitat Autònoma de Barcelona, 08193 Bellaterra, Catalonia, Spain

Correspondence should be addressed to Cristina Pérez-Solà; cristina.perez@uab.cat

Received 18 May 2017; Accepted 3 December 2017; Published 1 March 2018

Academic Editor: L. J. García Villalba

P2P networks are the mechanism used by cryptocurrencies to disseminate system information while keeping the whole system as much decentralized as possible. Cryptocurrency P2P networks have new characteristics that propose new challenges and avoid some problems of existing P2P networks. By characterizing the most relevant cryptocurrency network, Bitcoin, we provide details on different properties of cryptocurrency networks and their similarities and differences with standard P2P network paradigms. Our study allows us to conclude that cryptocurrency networks present a new paradigm of P2P networks due to the mechanisms they use to achieve high resilience and security. With this new paradigm, interesting research lines can be further developed, both in the focused field of P2P cryptocurrency networks and also when such networks are combined with other distributed scenarios.

1. Introduction

Since 2009, when the Bitcoin cryptocurrency [1] was released, a plethora of more than 600 different cryptocurrency proposals have appeared. The market capitalization of cryptocurrencies surpasses 51 billion US dollars (data retrieved from http://coinmarketcap.com/ on May 8, 2017) with Bitcoin leading the market as of today.

Security and robustness are probably the most important properties for a currency, and cryptocurrencies achieve them by using cryptographic techniques and a decentralized approach. Decentralization avoids both a single point of failure and a single trust anchor but potentially introduces discordances between parties. In order to reach consensus among nodes, cryptocurrencies take advantage of a distributed mechanism that allows the system to maintain a single unambiguous view of its state [2], the blockchain.

To support the communication between different entities of a cryptocurrency, a decentralized P2P approach is adopted to deploy the so-called P2P cryptocurrency network, that is, the communication overlay that transports all data needed in the cryptocurrency system. The main goals of such a network are, firstly, to allow members of the network to synchronize their view of the system state and, secondly, to disseminate peer information in order to allow peers to reenter the system after a disconnection.

Although the goals of P2P networks are shared among all blockchain-based cryptocurrencies, there is no standard for P2P cryptocurrency networks. In this paper, we analyze the Bitcoin P2P network to characterize general P2P cryptocurrency networks. Two main reasons made us choose Bitcoin as the subject of analysis. On one hand, far beyond the economic impact, being the largest cryptocurrency also conveys technical implications: both the volume of information flowing through its network and its size and heterogeneity surpass any other deployed cryptocurrency. On the other hand, being Bitcoin the first open-source cryptocurrency proposed, other new cryptocurrencies are developed as a software fork of the Bitcoin reference implementation. Although new cryptocurrencies have tweaked the Bitcoin source code in order to achieve different properties, an in-depth analysis shows that network mechanisms are usually unmodified and, in fact, even multiple cryptocurrencies share exactly the same network behavior as Bitcoin [3].

The first objective of this paper is twofold. On one hand, the paper fully describes the Bitcoin P2P network. On the

other hand, it characterizes the network to show how the aforementioned network goals, together with the special format of the information being transmitted through the network, conform to a new paradigm for P2P networks. This characterization will point out how, when considering cryptocurrency P2P networks, some of the well-known problems of P2P networks are not a concern, while other problems pose entirely new challenges.

The second objective of this paper is to analyze to what extent the adoption of cryptocurrencies, and their underlying P2P networks, can be a powerful tool for the development of distributed applications with mobile components. There are three relevant properties of cryptocurrencies that can be used as building blocks for such applications: secure distributed payment mechanisms, distributed storage with integrity by design, and secure transfer and distribution of digital assets. We analyze how these properties can be used to support distributed applications such as mobile crowdsensing or distributed IP/name resolution, to cite just two examples.

The structure of this paper is the following. First of all, in Section 3 and preceded by a basic description of the Bitcoin system, we provide a global description of all the elements in the Bitcoin P2P network, an overview that, to our best knowledge, lacked in the scientific literature (the only reference we are aware of is [4], and it is mainly focused on the economic aspects of the Bitcoin network). Second, in Sections 4 and 5, we perform a deep analysis of the Bitcoin network, which is compared to other existing P2P paradigms through a well-known P2P taxonomy. This characterization allows us to provide enough evidence to show that P2P cryptocurrency networks represent a new paradigm for P2P networks. Finally, in Section 6, we identify different applications in the field of mobile computation where cryptocurrencies may be applied, and we point out some of the opportunities and challenges that such an interaction may entail.

2. A Basic Description of the Bitcoin System

In this section, we point out the main ideas to understand the basic functionality of the Bitcoin cryptocurrency. Such a background is needed to understand the underlying P2P network that supports the communication between Bitcoin entities. However, the complexity of Bitcoin makes it impossible to provide a full description of the system in this review, so interested readers can refer to Narayanan et al.'s book [2] for a detailed and more extended explanation on the Bitcoin system.

Bitcoin is a cryptocurrency based on accounting entries [5]. Therefore, bitcoins should not be seen as digital tokens but as the balance of a Bitcoin account. A *Bitcoin account* is defined by an elliptic curve cryptography key pair. The Bitcoin account is publicly identified by its *Bitcoin address*, obtained from its public key. Using this public information, users can send bitcoins to that address (notice that the terms "public key," "address," or "Bitcoin account" refer to the same concept). Then, the corresponding private key is needed to spend the bitcoins of the account. Special purpose software, commonly referred as *wallets*, has been developed to create and manage those private keys and addresses.

Payments in the Bitcoin system are performed through transactions between Bitcoin accounts. A *Bitcoin transaction* indicates a Bitcoin movement from source addresses to destination addresses. Source addresses are known as *input addresses* in a transaction, and destination addresses are named *output addresses*. As it can be seen in Figure 1, a single transaction can have one or multiple input addresses and one or multiple output addresses.

A transaction (implicitly) details the exact amount of bitcoins to be transferred from each input address. The same applies to the output addresses, indicating the total amount of bitcoins that would be transferred to each account (although in this case, the specification is explicitly made). The Bitcoin protocol forces input addresses to spend the exact amount of a previously received transaction (notice that, in Figure 1, there are two input addresses that are exactly the same, which indicates that bitcoins have arrived to this Bitcoin account in two separate transactions). Therefore, each input must unambiguously indicate the previous transaction identifier (a transaction is identified in the Bitcoin system by its hash value) and the index of the output where the bitcoins were received. As a consequence, at any given moment, an output may be in two states: either already spent or not yet spent. An output that has not been spent is known as unspent transaction output, or *UTXO*.

Finally, the owner of the input addresses should perform a digital signature using his private keys to authorize a Bitcoin transfer, proving that he is the real owner of such accounts (although this is the standard form of Bitcoin verification for regular Bitcoin transfer transactions, the verification of a transaction can be much more complex and is based on the execution of a stack-based scripting language (more details can be found in Chapter 3 of [2])).

Before accepting a payment from a standard transaction, the receiver should

(i) validate that the digital signatures are correct;

(ii) validate that the bitcoins of the input addresses are not previously spent.

The first validation can be performed with the information included in the transaction itself (field *ScriptSig*) together with the information of the transaction identified in the *Previous output (Index)* (field *scriptPubKey*). The second validation prevents double-spending in the Bitcoin system, and it is performed through a ledger, the blockchain, where all previous transactions are annotated.

The *blockchain* is a general append-only ledger containing all Bitcoin transactions performed since the system started to operate (back in 2009), and it is freely replicated and stored in different nodes of the Bitcoin network, making the Bitcoin a completely distributed system.

Transactions are included in the blockchain at time intervals, rather than in a flow fashion, and such an addition is performed by collecting all new transactions of the system, compiling them together in a data structure called block, and including the block at the top of the blockchain. Every time that a block containing a specific transaction is included in the blockchain such a transaction is said to be a *confirmed*

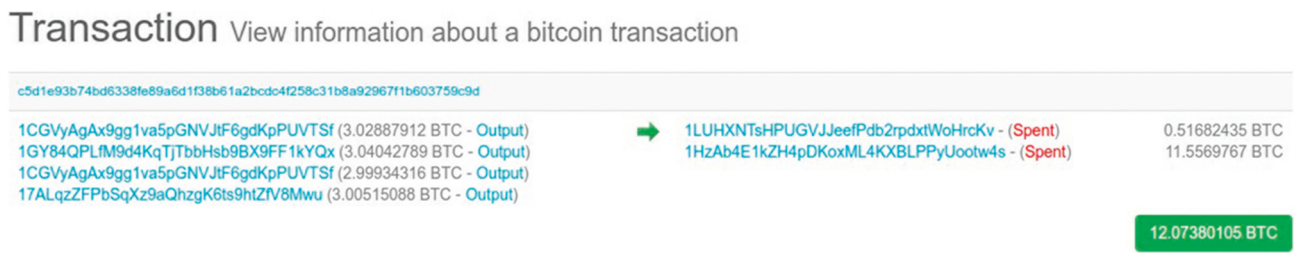

Figure 1: Bitcoin transaction example: four input addresses and two output addresses (data from http://blockchain.info).

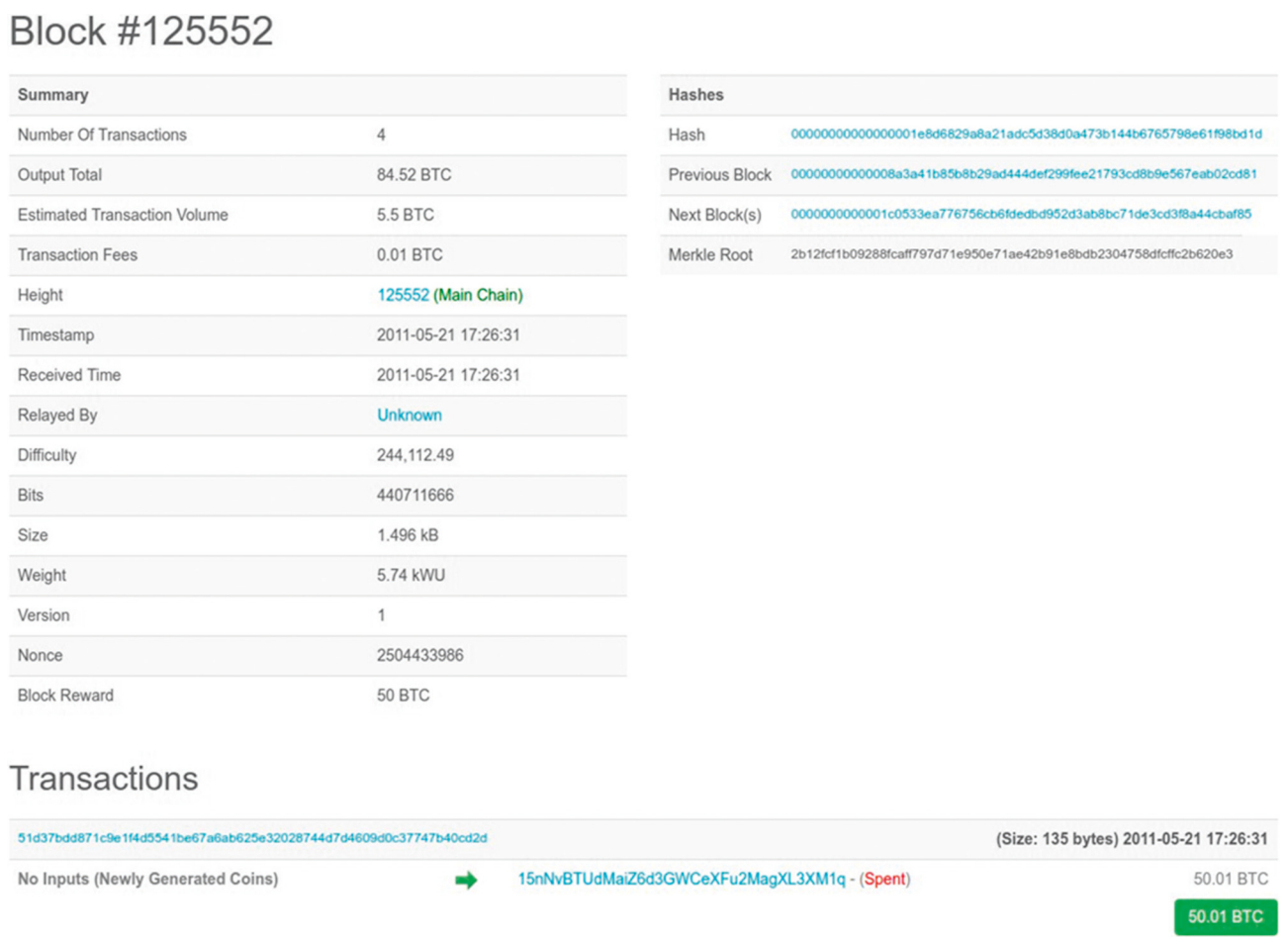

Figure 2: Example of a Bitcoin block (data from http://blockchain.info).

transaction since it has already been included in the blockchain and can be checked for double-spending prevention.

Blocks are data structures that mainly contain a set of transactions that have been performed in the system (Figure 2). To achieve the append-only property, the inclusion of a block in the blockchain is a hard problem, so adding blocks to the blockchain is time- and work-consuming. Furthermore, every block is indexed using its hash value, and every new block contains the hash value of the previous one (see the field *Previous block* in Figure 2). Such a mechanism ensures that the modification of a block from the middle of the chain would imply to modify all remaining blocks of the chain from that point to the top in order to match all hash values.

Adding a block to the blockchain is known as the *mining process*, a process that is also distributed and that can be performed by any user of the Bitcoin network using specific-purpose software (and hardware). The mining process uses a hashcash proof-of-work system, first proposed by Back as an antispam mechanism [6]. The proof of work consists of finding a hash of the new block with a value lower than a predefined target (notice that the value of the target determines the difficulty of the mining process. Bitcoin adjusts the target value depending on the hash power of the miners in order to set the throughput of new blocks to 1 every 10 minutes (in mean)). This process is performed by brute force varying the nonce value of the block. Once the value has been found, the new block becomes the top block of the

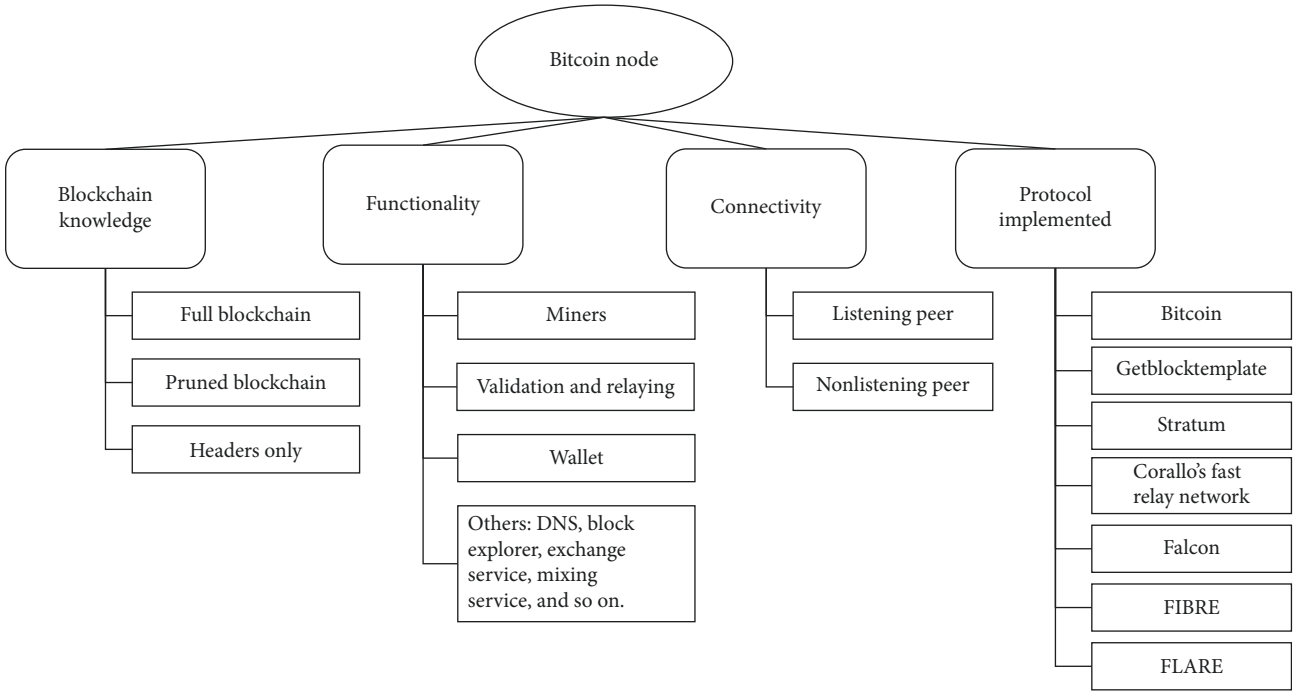

Figure 3: Bitcoin node classification.

blockchain, and all miners discard their work on that block and move to the next one.

Mining new blocks is a structural task in the Bitcoin system since it helps to confirm the transactions of the system. For that reason and also assuming that mining implies a hard work, miners have to be properly rewarded. In the Bitcoin system, miners are rewarded with two mechanisms. The first one provides them with newly created bitcoins. Every new block includes a special transaction, called *generation transaction* or coinbase transaction (see the first transaction in Figure 2), in which it does not appear to have any input address and the output address is determined by the miner who creates the block, who obviously indicates one of its own addresses (the amount of a generation transaction is not constant, and it is determined by the Bitcoin system. Such a value, started in 50 bitcoins, is halved every four years, fixing asymptotically to 21 million the total number of bitcoins that will ever be created).

The Bitcoin system needs to disseminate different kinds of information, essentially, transactions and blocks. Since both are generated in a distributed way, the system transmits such information over the Internet through a P2P network, that we describe in detail in the next section.

3. Description of the Bitcoin P2P Network

Bitcoin was first presented to the public in a white paper [1] describing its main concepts. Some months later, an open-source implementation of the Bitcoin client was released, giving birth to the cryptocurrency we now know and the P2P network that supports it. Such P2P network definition and implementation have been cloned in multiple new cryptocurrencies that derive from the Bitcoin implementation. In such new cryptocurrencies, the network configuration has been implemented almost identically. For instance, as described in [3], Litecoin, Dogecoin, Dash, and Peercoin have exactly the same network message types of Bitcoin, being the resulting networks for those cryptocurrencies very similar and in some cases identical to the Bitcoin one.

Since its deployment in 2009, where the only Bitcoin client available was the reference client, the Bitcoin network is now made up of very heterogeneous peers, whose hardware capabilities and software implementations differ largely from each other. Moreover, even new protocols have been created with the goal of optimizing certain tasks the Bitcoin ecosystem needs.

In order to describe the existing Bitcoin network, we first identify some of the properties that characterize Bitcoin peers. After that, we review the most common peer configurations, using the properties described before. Finally, we describe the composition of the current Bitcoin network.

3.1. Properties Describing Bitcoin Nodes. As we have already mentioned, Bitcoin peers are heterogeneous, presenting notable differences in both their hardware and software. In this section, we focus on describing the main properties that define a Bitcoin node: the exact part of the blockchain stored, its main functionalities, its connectivity, and the protocols it uses to communicate with other nodes. Figure 3 summarizes such a classification.

Peers participating in the network store some data about the *blockchain*. However, the exact data they store differ largely, from a few megabytes to dozens of gigabytes. *Full blockchain* peers store a complete and up-to-date version of the blockchain (on September 2016, the total size of block

headers and transactions consists of 80 GB of data). *Pruned* blockchain peers store an up-to-date version of the blockchain with complete blockchain data for at least the last 2 days (the number of days for which to store complete blockchain data can be tuned by users. Pruned mode is estimated to reduce disk usage to around 2 GB [7]). Although only storing complete blockchain data for a few days, pruned nodes are able to securely validate transactions because they indeed store the required information from their previous history of the blockchain, that is, metadata about all known blocks and the UTXO set. *Simplified payment verification (SPV) clients* have an up-to-date version of the blockchain headers (a block header is an 80-byte structure. On September 2016, the Bitcoin blockchain has 432,000 blocks, thus needing around 33 MB of the disk space). Additionally, SPV clients may store transaction data from some transactions of interest. SPV clients are usually deployed in mobile devices such as smartphones, where having the full blockchain is generally unaffordable.

Peers can also be classified on the basis of their *functionality*. There are three functionalities needed for the Bitcoin system to work. *Mining* is the computationally expensive task of trying to create blocks. New blocks are appended to the end of the blockchain, thus making the public ledger grow. Peers that perform mining are known as miners. Some peers perform *validation and relaying* of the transactions and blocks they receive, that is, they relay to other peers valid transaction and block data, together with network data. Some peers also have a *wallet* functionality, that is, they store a set of key pairs, they track the amount of bitcoins deposited on addresses associated with those keys, and they are able to create transactions that spend those bitcoins. These functionalities do not necessarily exclude each other, that is, a peer may perform more than one functionality at the same time. Additionally, although not strictly necessary for Bitcoin to work, some peers may provide other functionalities. For instance, they may provide a *DNS* service, that offers information about existing peers; a *block explorer service*, where it is possible to query for transaction and block data through a graphical interface; an *exchange* service, where users can buy or sell bitcoins in exchange for other currencies; and *mixing services*, where users are able to obfuscate the history of their coins.

Depending on their *connectivity*, peers can be classified into *listening* peers or *nonlistening* peers. Listening peers are nodes that accept incoming connections, while nonlistening nodes are those not doing so. Although most Bitcoin full implementations listen for incoming connections, some network configurations do not allow these connections to be created (e.g., peers behind NAT).

Even though the original Satoshi Bitcoin paper implicitly assumed that peers would use only one *protocol*, the Bitcoin economy has grown much bigger than the original specification, giving place for lots of protocols to arise. We will use the term "*Bitcoin* protocol" to refer to the network protocol used by the current standard implementation, the Satoshi client. Other protocols that currently exist on the Bitcoin system are mainly targeted to optimize pooled mining and speed up data propagation. *Getblocktemplate* is the new Bitcoin pooled mining protocol (supersedes the previous mining protocol *getwork*), where the full block data are sent to miners. This allows miners to change the content of the block by themselves, thus gaining autonomy with respect to the pool servers. *Stratum* is a protocol first designed for lightweight clients and later extended to handle pooled mining. With respect to mining, it does not send full blocks to miners, thus better scaling with the number of transactions but providing less autonomy to miners to decide what to include in the blocks. The *Bitcoin relay network* has a protocol for communicating with Corallo's fast relay network backbone, a 6-node network intended to speed up the relaying of Bitcoin data. Similarly, *Falcon* is also a backbone of nodes intended to make Bitcoin data propagation faster. Peers can connect to Falcon using either the Bitcoin protocol or a specially designed network protocol that relays packets as received (instead of waiting for all packets of a full block to be received before starting to relay that block). Again with the purpose of speeding up the block propagation, *FIBRE* (Fast Internet Bitcoin Relay Engine) is a protocol that uses UDP with forward error correction to decrease the delays produced by packet loss. It also introduces the usage of compression to reduce the amount of data sent over the network. The lightning network is arising as one of the solutions to Bitcoin scalability limitations. In this context, *FLARE* is the new proposal for a routing protocol for the lightning network.

3.2. Archetypal Bitcoin Nodes. The term "*full client*" is used to define peers that perform full validation of transactions and blocks. In order to perform this full validation, they need to store either the full blockchain or a pruned version. There currently exist many implementations of full clients.

The reference implementation of Bitcoin is known as the Satoshi client, which is currently used to refer to both the *Bitcoin core* and *bitcoind*. Bitcoin core provides a graphical interface, whereas bitcoind is intended for RPC use and does not have a graphical interface. Currently, the Satoshi client is a thick client that may work either with the full blockchain (this is currently the default option) or with a pruned version. It used to have mining functionalities incorporated, but one of the latest versions [8] removes the internal miner and leaves just a minimal functionality for testing purposes. The Satoshi client performs validation and relaying of blocks and transactions and provides a basic wallet. It serves as a reference for the Bitcoin protocol and also incorporates the Bitcoin mining protocol Getblocktemplate. The software tries to create outgoing connections to the P2P network and also listens for incoming connections from other peers.

In early 2016, there was a vivid debate about a change in the consensus rules to increase the block size limit. From that debate, three different forks of the Satoshi client appeared (which maintain the original properties but change the consensus rules regarding the block size limit). *Bitcoin classic* increased the block size limit to 2 MB. *Bitcoin XT* changed the limit to 8 MB (with subsequent increases over time). *Bitcoin Unlimited* proposed to remove the limit.

Apart from the implementations that appeared from the disagreements on how to handle block size limitations, other

Table 1: Properties of archetypal Bitcoin nodes. The properties described in the table refer to the most common nodes of each type, but due to the vast heterogeneity of Bitcoin nodes, some differences may be found in the real network.

Node	Blockchain	Functionality	Connectivity	Protocol
Full client	F/P	V/R, W	L/NL	B
SPV client	H	W	NL	B
Non-SPV light client	—	W	—	S/SP
Solo miner	F/P	V/R, W, M	L/NL	B
Pool mining server	F/P	V/R, W, M	L/NL	B/S/SP
Pool mining client	—	W, M	—	S/SP

forks from the Satoshi client currently exist, for instance, *Bitcoin Knots*. There also exists some implementation of full clients that are not forks of the Satoshi client. For instance, *Bitcore* (Javascript), *bitcoinj* (Java), or *btcd* (Go).

SPV clients are peers that only have a full copy of the blockchain headers, which allows them to save on space requirements. However, they are not able to perform full validation of transactions and blocks since they lack the needed information to do so. Their main functionality is as wallets. To that end, apart from the blockchain headers, they also store cryptographic keys that allow them to spend bitcoins and the transactions that are related to those keys. SPV clients use the Bitcoin SPV protocol.

There exist many implementations of SPV clients, for instance, *breadwallet*, *Electrum*, *Bither*, *GreenBits*, *Simple Bitcoin*, *Bitcoin Wallet*, or *MultiBit HD*.

SPV clients are said to be lightweight clients because they minimize the resources needed to accomplish their functionality. However, there exist *other lightweight clients* that are not based on SPV. The current alternatives are centralized approaches, where clients connect to a set of predefined servers that relay them the information they need in order to work as wallets. This approach requires to trust the servers. The specific amount of data about the blockchain or cryptographic keys stored by these clients depends on each implementation. Some of them publish their source code for public review, while others do not. The protocol is also specific. Some examples of these kinds of wallets are *Mycelium*, *Coinomi*, *Coin.Space*, or *Copay*.

Solo miners are peers whose main functionality is mining. Initially, they had a full copy of the blockchain (or at least a pruned copy), in order to be able to validate the transactions they include in blocks, and they communicated using the Bitcoin protocol. They also needed a wallet in order to manage their mining rewards. However, as mining has become more and more specialized with the introduction of dedicated hardware, the paradigm has changed, and currently, the mining task is split into two: block structure creation and hashing. The first task is performed by peers that do have a copy of the blockchain and validate the transactions they include in blocks, whereas the second task is performed in specialized hardware, optimized to speed up hashing.

Moreover, lots of miners group together in order to reduce the amount of redundant work and to minimize the variance of the rewards obtained from the mining process. Groups of miners are known as pools and usually operate as client-server architectures, with the pool operator providing a *pool mining server* to which *pool mining clients* connect to in order to retrieve their portion of work. Clients do not need to have any knowledge about the blockchain nor to perform any validation on transactions. Clients communicate with the pool server via specifically designed protocols such as stratum.

Some well-known software implementations for client miners are cgminer or *BFGMiner*. A basic miner server is included in *bitcoind*, and some existing complete mining servers are *CoiniumServ*, *ecoinpool*, or *Eloipool*.

Table 1 summarizes the properties of the aforementioned archetypal Bitcoin nodes. Regarding blockchain knowledge, F stands for full blockchain, P for pruned, and H for headers only. With respect to functionality, W means *wallet*, M mining, and V/R validation and relaying. Concerning connectivity, L means listening, while NL stands for non-listening. Finally, as regards to the protocol, B stands for Bitcoin, S for stratum, and SP for specific protocols.

3.3. Network Description. In order to better characterize the so-called Bitcoin network, let us define three subsets of the overall network, as represented in Figure 4:

(i) The *reachable* Bitcoin network is composed of all listening nodes that talk the Bitcoin protocol. The size of the reachable Bitcoin network is estimated to be in the range of 5,000 to 10,000 nodes [9].

(ii) The *nonreachable* Bitcoin network is made of nodes that talk the Bitcoin protocol, regardless of whether they are listening for incoming connections. The size of the nonreachable Bitcoin network is estimated to be 10 times bigger than that of the reachable Bitcoin network.

(iii) The *extended* network comprises all nodes in the Bitcoin ecosystem, even those not implementing the Bitcoin protocol. This network includes, for instance, pooled miners communicating with the pool server using only the stratum protocol. To our best knowledge, there are no estimations on the number of nodes that belong to the extended network.

Both the *reachable* and *nonreachable* Bitcoin networks are P2P networks: they are distributed systems built without mediation of a centralized server or authority, they can adapt to changes in the network and their participants autonomously, and their nodes contribute to storage, computing power, and bandwidth to the network.

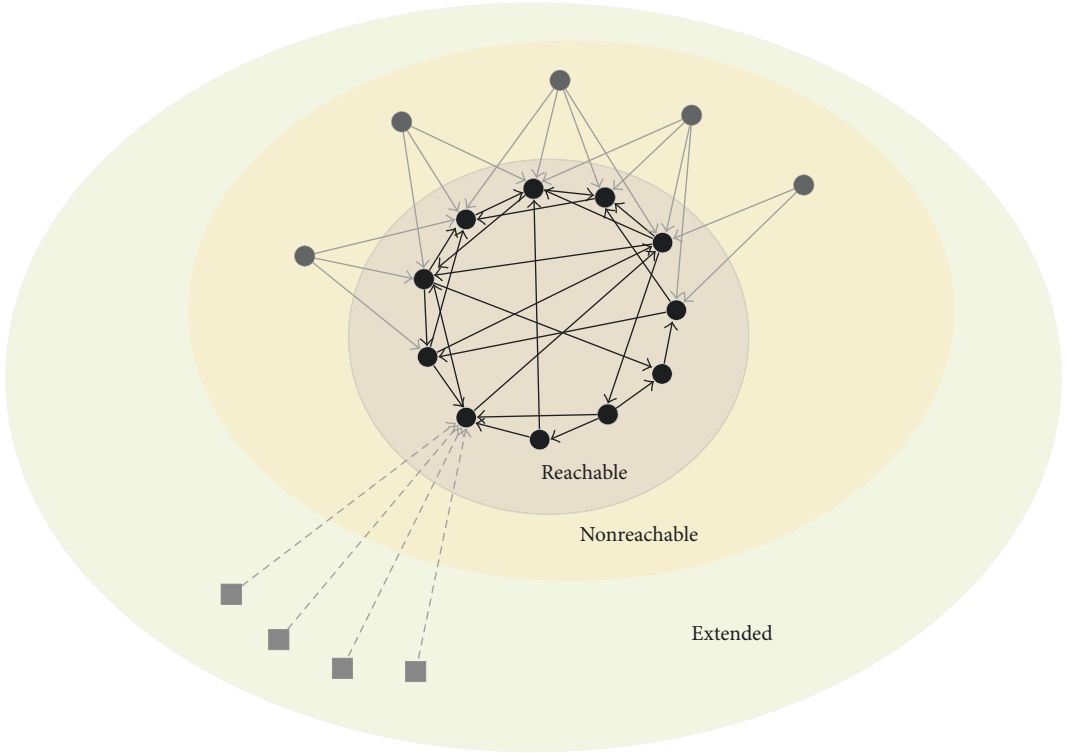

Figure 4: Abstraction of the defined network subsets: nodes in the *reachable* network accept incoming connections; nodes in the *nonreachable* network just create outgoing connections; nodes in the *extended* network do not implement the Bitcoin protocol (for instance, miner clients using the stratum protocol).

In this section, we have provided a detailed description of the Bitcoin network by first describing the main properties that define a Bitcoin peer, then identifying the most common Bitcoin peers, and finally providing an overview of the network. Having described the Bitcoin network, the next section provides its characterization as a P2P network.

4. Network Characterization

In order to characterize the new P2P network paradigm that cryptocurrency networks represent, we perform an analysis of the Bitcoin P2P network using the taxonomy defined by Lua et al. [10] for the comparison of different P2P overlay network proposals. Following the same taxonomy, we will be able to stress the differences of such new networks in comparison with the existing ones. The following analysis is performed aiming only at the Bitcoin reachable network, following the classification established in Section 3, since it is the only full P2P part of the Bitcoin network.

4.1. Decentralization. Decentralization assesses to what extend the analyzed network presents a distributed nature or, on the contrary, its configuration shows some centralized characteristics. In that sense, the Bitcoin network is a nonstructured P2P overlay with some similarities with Gnutella. With a flat topology of peers, in the Bitcoin network, every peer is a server or client, and the system does not provide centralized services nor information about the network topology.

4.2. Architecture. The architecture describes the organization of the overlay system with respect to its operation. As we already indicated, the Bitcoin network presents a flat architecture with no layers nor special peers. The network is formed by peers joining the network following some determined basic rules, where randomness is an essential component. Such a random behavior in the network creation intends to generate an unpredictable and uniform network topology, unknown to its users. As we will see in Section 5, such lack of knowledge about topology is needed for security reasons.

4.3. Lookup Protocol. One of the main problems in P2P networks, specially those used for content distribution, is the lookup query protocol adopted by the overlay to find the desired content. However, although the Bitcoin network can be regarded as a content distribution network (where content is transactions and blocks), the information flowing in the network is completely replicated in every node. Hence, there is no need for such a lookup protocol since information is always available at one hop peer at most. However, information propagation has to be performed in order to synchronize all peers of the network with the same data. Such a propagation is performed through the controlled flooding protocol.

Mainly, the controlled flooding protocol works on a push paradigm, propagating the data as they are generated. Instead of being directly sent, data availability is announced to the selected peers, and in case a peer lacks some of the announced information, he requests it back to the announcer. Two types of data structures are propagated through the network in that way: *transactions* and *blocks*.

4.3.1. Transaction Propagation. Transactions are the basic data structure flowing though the Bitcoin network and the one most usually seen. Every single node can take part in a transaction by simply using a wallet, no matter of its type. Transactions flow though the network aiming to reach every single node to, eventually, be included in a block. The average number of transactions per day during September 2017 was 222,458 moving around 1,808,009 bitcoins between different accounts (raw data obtained from https://blockchain.info/charts).

4.3.2. Block Propagation. Blocks are the data structure the blockchain is built from and include some of the transactions that have been created during the block mining process. Unlike transactions, blocks require a tremendous hashrate to be generated, which virtually limits their creation to mining pools. Moreover, the block generation throughput is set by design to 6 blocks per hour, periodically adjusting the block mining difficulty according to the total network hashrate.

Nonetheless, a pull data synchronization mechanism is also performed in the network, and while having a quite specific use, it is fundamental for its proper operation. Its main purpose is to synchronize the blockchain of outdated nodes, that have been off-line when data have been propagated. Outdated nodes request an on-demand synchronization to their peers during the bootstrapping phase, obtaining all the missing blocks in their local blockchain. Such a request does not refer to specific block values but to all blocks above the last block the enquirer is aware of. Besides blocks, on-demand propagation of other types of data, such as transactions, is not set by default. Only nodes that have built a full index of transactions along the blockchain, like *block explorer services*, can provide this type of data since normal nodes only track transactions bounded to their addresses.

4.4. System Parameters. Different P2P network overlays require a set of system parameters for the overlay system to operate. For instance, structured P2P networks require to store information on the distribution of peers in the network in order to improve routing performance. However, the Bitcoin P2P network, in line with other unstructured P2P overlays, does not require any special system parameters for the normal behavior of the network. Every single node could join the network with no prior knowledge of it. Apart from that, some default parameters are used by nodes, such as the maximum connection limit set to 125, although such a value is not a restriction and each node could select the number of connections it wants to maintain.

4.5. Routing Performance. Differing from traditional P2P networks (such as Gnutella [11]), Bitcoin does not follow a multihop routing scheme. As already pointed out before, peers in the network store a replica of all the information that has been flowing through the system up to the date, namely, the blockchain. In that way, no queries are forwarded between peers since all information is supposed to be located at one hop peer at most. Therefore, data are guaranteed to be located if the network is synchronized, and no routing protocol is needed nor used, apart from the synchronization protocol.

Propagation delay is therefore a fundamental factor for the Bitcoin network in order to achieve synchronization at any given time. In that way, Decker and Wattenhofer [12] analyzed the block propagation time for 10,000 blocks and discovered that it followed an exponential curve, being the median block propagation time 6.5 seconds while the mean was located at 12.6 seconds. However, the distribution showed a long tail, implying that a short percentage of the nodes (5%) required more than 40 seconds to receive the blocks. Furthermore, an analysis of how block size influences on the propagation delay was also performed. They reach the conclusion that, for small amounts of data, that is, less than 1 kB, there is a huge overhead during the propagation since the protocol involves multiple messages while negotiating the information that has to be forwarded. This applies basically to transactions, actually to a huge amount of them (96%), while not to blocks. For data size larger than 20 kB, the added overhead is negligible (around 80 ms).

4.6. Routing State. Despite being a content distribution network, the routing state of Bitcoin cannot be directly defined due to the randomness and dynamism of its topology and to the fact that it is not known. Moreover, as we have pointed out before, no multihop routing is performed since data could be found at one hop peer at most.

4.7. Peers Join and Leave. How to build the network is a classic problem P2P networks have to deal with. From building the network from its roots to how nodes deal with peer disconnection, P2P networks need to be highly adaptable to avoid partitioning. In order to deal with this problem and also provide a fair and secure way to choose the peers a node is going to be connected to, the Bitcoin network performs a particular network discovery mechanism.

By default, all peers maintain up to 125 connections with other peers. Each node will start 8 of those connections with other peers (namely, outgoing connections) and will accept up to 117 from potential peers (namely, incoming connections). Despite the name, all connections are bidirectional. In order to pick the outgoing connections, every single node will look for a subset of nodes it stores in a local database. This database is formed by two different tables: *tried* and *new*. *Tried* table contains addresses from peers the node has

already connected to, and *new* table contains addresses the node has only heard about. Additionally, when the node tries to establish a connection to the network for the first time, it queries a well-known list of DNS nodes, that will provide a set of online potential peers (further information about how peers are stored and selected can be found in [13]).

Nodes try to always maintain their 8 outgoing connections, selecting new peers from the database if any of the established connections is dropped. Peers are stored and selected from the database following a pseudorandom procedure that gives the network high dynamism and keeps its structure unknown. Peer information can be obtained by a node following two ways. First of all, a node could request such data to its neighbors, in order to fill up its database, through sending a *getaddr* message, or could receive such information spontaneously from one of its peers without any kind of request. In both cases, the information is sent using a set of *addr* messages, containing up to 2,500 peer addresses both from the neighbor's *tried* and *new* tables. Such addresses are stored in the local node's *new* table. On the other hand, an *addr* message containing a single address could be sent to a node when a node wants to start a connection with a potential peer. By sending its address, the node notifies the receiver that it has been picked as a peer, and if the latter has room for more incoming connections, the communication is established. Peer addresses received in that way are stored in *tried* table. All addresses are stored in the database together with a timestamp that helps the node to evaluate the freshness of such an address when selecting a peer.

4.8. Security. Security in P2P networks has always been a broad topic since multiple security threats can be identified in different P2P implementations. The interested reader can refer to Wallach's survey [14] for an introduction to the topic of security in general P2P networks, to Bellovin's paper [15] for a description focused on the security issues affecting specific P2P protocols such as Napster and Gnutella, and to [16] for an introduction to security problems in P2P SIP communications.

However, in P2P cryptocurrency networks, security takes a different twist. At first sight, one could believe that the threats P2P cryptocurrency networks face are a subset of the threats found in standard P2P networks. However, as we will see in detail in the next section, most of the threats encountered in general P2P networks do not apply directly to P2P cryptocurrency networks due to the cryptographical mechanisms used by the currencies and the level of security offered by their protocols.

Additionally, one can also believe that multiple new threats will also arise in cryptocurrencies due to the sensitivity they have as money transfer networks. However, as we will see in the next section, this is not also the case.

In the next section (Section 5), we provide a detailed review of the most common security threats identified for typical P2P networks and discuss to what extent they affect the Bitcoin network.

4.9. Reliability and Fault Resiliency. Reliability and fault resiliency analyze how robust the overlay system is when subjected to faults. Typically, such robustness measurements are related to nonintentional failures, for instance, by a massive disconnection of peers of the network or an increasing volume of information being transferred through the network, but do not include intentional attacks that would be categorized inside the security properties of the network.

Bitcoin implements a distributed consensus protocol resilient to Byzantine faults. That is, the protocol is resistant to arbitrary faults produced in the participating peers, from software errors to adversary attacks. The main idea behind this protocol is to use a proof-of-work system to build the public ledger where transactions are stored. Appending new information to the public ledger requires a huge amount of computer power, thus preventing attackers to monopolize ledger expansion and censuring transactions. In a similar way, changing the content of the blockchain is also computationally expensive, up to the point that transactions are considered secure when they have 6 confirmations (i.e., five blocks have been created on the top of the block that included the transaction). Additionally, the blockchain is replicated on all full blockchain nodes, contributing to the fault resiliency of the system and providing high availability of the ledger data.

Assuming that categorization, the Bitcoin P2P network has been designed with a high level of reliability, thanks to the redundancy that implies the storage of all the relevant information of the network in every peer of the network. With this approach, the high inefficiency level in terms of storage space is translated into a high resilience of the network since the availability of a single node in the network contains the information to keep the system alive. Moreover, the proof-of-work system allows peers to (eventually) reach a consensus state, even in the presence of attackers trying to subvert the system. As a drawback, the consensus protocol is somehow slow, with transactions needing 9 minutes (median confirmation time as of October 13, 2016 [17]) to confirm, and expensive, requiring the consumption of lots of energy for each mined block.

5. Security Concerns in P2P Networks

Security in P2P networks has been extensively studied in the literature. In this section, we provide a broad overview of the main security problems that arise in P2P networks, we review how each of the security problems may affect the Bitcoin network, and if it is the case, we explain the specific countermeasures Bitcoin provides in order to defend from each attack.

The list of reviewed attacks goes over the most typical types of attacks and security flaws found in common P2P networks. It is clear that specific networks and applications might present specialized attacks, but in most cases, they can be seen as a specification of the attacks presented here.

So as to provide a clear picture of how common P2P attacks affect Bitcoin, we first review the three attacks that have been shown to be clearly applicable to Bitcoin. After that, we include a list of attacks identified for common P2P networks, but this does not have such a high impact on Bitcoin, reviewing why the attacks do not apply to the specific

Bitcoin network and detailing the particular cases where those attacks (or some variation) may somehow relate to Bitcoin.

In favor of a clear and concise presentation, we have not explicitly covered some recent attacks such as [18], which do not directly affect or involve the Bitcoin network, or network-related attacks such as [19], which rely on BGP hijacking and are thus out of the scope of our study.

5.1. DoS Flooding. Denial-of-service (DoS) attacks are possible in most P2P scenarios and are especially relevant, for example, in P2P streaming applications [20–22]. Given their dynamic nature, P2P networks are usually more resilient against generic DoS attacks than more static networks. Targeted DoS attacks to specific parts of the P2P network (a given node) or services are usually more important.

There exist several potential DoS flooding attacks in Bitcoin, but the system has countermeasures in place. *Transaction flooding* is prevented by not relaying invalid transactions and imposing fees to valid transactions. On one hand, transactions are signed by the senders in order to demonstrate that they are authorized to transfer those bitcoins. If the signatures of a transaction are not correct, the transaction is considered invalid and is not relayed to the network. On the other hand, the default protocol does not relay transactions without fees (except for a few very specific cases that would also result in very expensive attacks). Moreover, transaction's fees increase for lower input ages (i.e., for bitcoins that have been moved recently), so an attacker trying to generate a huge amount of transactions that move the same bitcoins would have to pay increasing fees. *Block flooding* is prevented by only relaying valid blocks, which must contain a valid proof of work. In order for a block to contain a valid proof of work, its hash must be lower than a given target. Obtaining a block with such a hash is a computationally expensive task, thus performing DoS attacks with block data unfeasible. *Network data flooding* is easier than the previous two cases because it is indeed possible to create valid network messages without paying fees nor spending computation cycles. However, Bitcoin has a banning protocol: peers may ban other peers for one entire day if their misbehavior score crosses a certain threshold. The misbehavior score is increased for sending duplicate version messages, sending large messages, and sending invalid blocks. Given the nature of Bitcoin, *cpu usage DoS* is possible by trying to make peers spend lots of time validating a transaction or a block. In order to prevent this kind of attacks, Bitcoin tries to catch errors before starting to validate a transaction, limits the number of signature operations per transaction and per block, and limits the size of the script. Finally, previous versions of the Bitcoin client were also susceptible to *continuous hard disk read attacks*, where an attacker repeatedly sent double-spend transactions that passed the initial checks and required to retrieve data from disk in order to be fully validated. This attack is now prevented by checking that the inputs of the transaction that is being validated are in the UTXO set (i.e., checking whether the transaction is a double spend) before retrieving any information from disk.

5.2. Eclipse Attacks. An eclipse attack occurs when an attacker creates (or has control of) a large number of distinct nodes that populate the whole neighborhood of the victim node [23]. The attacker can then *eclipse* the view of the network that has the victim. Common solutions for sybil attacks are usually insufficient to defend against eclipse attacks [24].

In a cryptocurrency network, isolating a node from the rest of the network may enable two other attacks to the eclipsed peer. First, an eclipsed peer may undergo a censorship attack because the victim's transactions must pass through the attackers' nodes in order to reach the network. Therefore, the attacker may decide not to forward these transactions, thus censoring the victim's transactions. Second, if the eclipsed victim is a miner, the attacker can drop or delay the propagation of the new blocks found by the rest of the network. As a consequence, the victim wastes computation time trying to mine on the top of old blocks.

Bitcoin has many defense mechanisms to prevent eclipse attacks, some of which were added recently, after a study pinpointed some of the flaws of the then vigent implementation [13]: the client restricts the amount of outgoing connections to addresses in the same network, randomizes the address selection procedure, and maintains a big list of peers, among others.

5.3. User Profiling. In some P2P networks, it is easy to record all the activities of a giving node allowing attackers to easily create identifying profiles of users and their activities. This is relevant in anonymous systems, or systems that want to guarantee a certain degree of anonymity [16, 25].

Bitcoin provides pseudonymity by allowing users to receive payments to their addresses, which are not initially linkable to their identities. The usage of new addresses for each transaction in the system is intended to provide unlinkability between the different actions a single user performs through Bitcoin. Therefore, user profiling in Bitcoin usually consists in attacking the unlinkability between different addresses a single user has. Three different approaches have been taken to perform address clustering: using network layer data [26], performing analysis over the transaction graph [27–29], and analyzing Bloom filters [30]. The idea of using network layer data to cluster addresses is straightforward: if an attacker is able to connect to all the peers of the network, the first node that sends him a given transaction should be the creator of that transaction. Therefore, if the attacker first receives two different transactions from the same peer, he can infer that the source addresses of both transactions belong to the same user. However, as simple the attack may seem conceptually, it is not that easy to perform in practice. It is not trivial to connect to all nodes of the network since most of them do not accept incoming connections. Moreover, some peers anonymize their connections using Tor. Finally, collected data are very noisy, and therefore, it is not easy to make strong claims when analyzing it. Regarding transaction graph analysis, there exist mixing services that are able to effectively break the relationship between an address and its

past. Additionally, the use of a secure wallet that tries to minimize the leaked information about addresses clusters helps mitigate the consequences of this kind of analysis. Finally, concerning the usage of bloom filters, users must be very careful when choosing the parameters of the filter and when generating different filters that match the same set of addresses and public keys. Additionally, new protocols are being designed to allow lightweight clients to retrieve their transactions of interest while maintaining privacy.

Bitcoin's scalability problems have triggered the search for new solutions that would allow to increase the transaction throughput of the network. Several proposals provide mechanisms to create off-chain payment channels, such that secure transactions between Bitcoin users may be performed without needing to include all the transactions into the blockchain. In turn, these solutions may also entail privacy problems that are yet to be carefully studied [5].

5.4. Other Attacks. After analyzing the three main attacks that have threatened the Bitcoin network over the last years, we summarize other common P2P attacks that have a lesser impact on Bitcoin. We will show how some of those attacks could be used as a preliminary phase to achieve one of the three previously introduced ones, while others are not harmful for the Bitcoin network due to its design.

5.4.1. ID Attacks. Two different subattacks can be identified in this category:

> *ID mapping attack*: when a node changes its own identifier with malicious purposes. As an example, in DHT-based P2P networks, a node can gain control over given resources by changing its own identifier [31]. These kinds of attacks are more difficult in networks where the identifier is derived from a public key [32].
>
> *ID collision attack*: similar to previous attacks, here the attack is considered to happen when there are duplicated identifiers. The problem is usually prevented by ensuring the uniqueness of identifiers [16].

There is no clear concept of a peer identifier in Bitcoin. Two different properties could be considered identifiers in Bitcoin, depending on the exact entity one wants to identify: IPs and Bitcoin addresses. IPs allow to identify peers, whereas addresses are linked to users. A malicious peer may benefit from a change of IP if it is banned for misbehavior. Each peer maintains a banscore for each of its neighbors. This banscore is increased whenever the peer misbehaves. If the banscore surpasses a certain threshold, the neighbor is banned for 24 hours. Therefore, being able to change the IP allows a peer to effectively reset its banscore. Regarding the second kind of identifiers, Bitcoin addresses, the recommended behavior for users is indeed to change them frequently. In fact, the suggestion is to not reuse addresses, that is, to create a new address for each transaction made in the system. This allows to protect user privacy.

5.4.2. Sybil Attack. A sybil attack is a well-known attack in P2P networks, where a malicious user creates multiple identities in order to control the system or parts of the system [33]. This has been very extensively studied in the literature in the context of several P2P technologies [34, 35].

Sybil attacks may be a problem in Bitcoin if they are able to eclipse all the connections from a peer (see Section 5.2 for details of eclipse attacks). However, besides its extension to an eclipse attack, a peer with multiple identities cannot harm the system regarding the main content of the network: transactions and blocks. Blocks cannot be counterfeit without the corresponding proof of work, and transaction generation entails an associated fee (in a similar way that was described in *flooding attacks* in Section 5.1). Nevertheless, if lots of sybil nodes start performing a huge amount of connections to the existing network, they may monopolize all available incoming connection slots, and the system decentralization could be reduced.

5.4.3. Fake Bootstrapping. Network access in P2P environments starts by connecting to one or multiple nodes of the network. This first contacted node is known as the bootstrap node. A malicious bootstrap node can influence the view of the network for the new user [23]. Several solutions already exist for this problem such as not relaying in a single bootstrap node, use of cached peers for subsequent connections, random address probing, using external mechanisms, using specific bootstrapping services, or using network layer solutions (e.g., use of a special multicast group for bootstrapping) [16, 36, 37].

Bitcoin deals with bootstraping issues by defining a local peer database on every single node, that is queried following a pseudorandom protocol to obtain a subset of potential peers (see Section 4.7 for details). In that way, Bitcoin applies most of the solutions for the fake boostrapping protocol, such as *not relaying in a simple bootstrap node*, by establishing 8 outgoing connections on every bootstrap, *use of cached peers for subsequent connections*, by using peers stored in *tried* table, *random address probing*, by using a pseudorandom protocol to store and retrieve peer addresses from the database, and *using external mechanisms* by quering a list of well-known DNS nodes or even using a list of hardcoded nodes, if the DNS cannot be reached.

5.4.4. Unauthorized Resource Access. P2P networks often use some sort of private data that have to be protected from unauthorized access. Common solutions are those typically employed for distributed access control [38, 39].

Bitcoin is based on public key cryptography, where private keys are needed to authorize payments. Therefore, private keys must be kept secret, and two methods are usually employed: encryption and off-line storage. By using encryption, private keys remain secure even if an attacker is able to retrieve the key file as long as the encryption key remains secret. As for off-line storage, different approaches can be taken with different technical sophistication levels, from the usage of dedicated hardware devices to paper wallets. Notice that unlike other uses of public key cryptography where private keys need to be online (for instance, in the handshake process in TLS), Bitcoin network operation

does not involve private information since validations are performed using public information. For that reason, offline storage of public keys does not impact the network performance.

5.4.5. Malicious Resource Management. A malicious node can deny the existence of a given resource under its responsibility, or claim to have a resource it does not have. This is specially relevant in content distribution applications, and common solutions are replication of resources [40], or use of error-correcting codes to reconstruct missing parts of the resource [41].

Bitcoin network is protected against malicious resource management by, on one hand, the high amount of data redundancy information of the network and, on the other hand, the multiple neighbors a node of the network is connected to. Thanks to the fact that peers establish connections (by default) to 8 other peers, if a given neighbor denies the existence of a certain resource, the peer can learn it from his other neighbors. Moreover, if a neighbor says he has some resource he actually does not have, peers will notice when they try to retrieve it (since transactions and blocks are identified by their hash).

5.4.6. Free Riding. A free-rider user or node in a P2P network is a node that attempts to benefit from the resources of the network (provided by other users) without offering their own resources in exchange [42, 43]. Depending on the application, this might not be an issue or even might not be considered a security problem. It is usually described in content distribution applications, and the main solutions proposed rely on incentive- or penalty-based mechanisms [44].

Bitcoin is sustained by an equilibrium of economic incentives. Miners are remunerated for their work by obtaining a reward for each block they successfully mine. Additionally, transaction senders (and, although indirectly, also transaction recipients) may include a fee to their transactions, which is also collected by the miner of the block that contains the transaction. As a consequence, miners are encouraged not only to create blocks but also to include transactions on those blocks. There is, however, a set of nodes whose role is important in ensuring the decentralization of the network and that do not directly receive economic incentives for their work: full clients. While these clients store the blockchain and perform validation and relaying of transaction and blocks, they do not get a direct economic reward in return for their work.

5.4.7. Man-in-the Middle (MITM). In the context of P2P networks, a MITM attack is usually considered a routing attack, similar to classical network MITM attacks. P2P networks, which require multihop routing, will need to include measures similar to onion routing in order to secure connections between all nodes along the path [45, 46].

MITM attacks in Bitcoin are not a problem for transaction and block integrity because transactions are cryptographically signed and blocks must contain a valid proof of work. Transaction malleability may be a problem (refer to Section 5.4.10 for a detailed explanation) in very specific scenarios, but a solution is currently being deployed. Censorship is neither a problem because a single peer maintains different connections. An attacker must be in the middle of all of them to hide information to the peer (thus resorting in *eclipse attacks*).

5.4.8. Replay Attack. A replay attack is produced when a legitimated transmission is delayed or lately replayed with malicious purposes. This is a very common network attack that can affect P2P networks in several ways but is usually solved at a protocol level.

Replaying transactions or blocks that have been sent to the network does not have any effect on the Bitcoin network. Nevertheless, delaying block propagation may be a beneficial strategy for miners [47]. By not immediately propagating a block the miner has just found, the miner can start working on top of this newly found block while making other miners lose time working on the previous block. This strategy is known as selfish mining and reduces the bound on the percentage of hashing power an attacker must have in order to successfully control the information appended to the ledger.

5.4.9. Routing Dysfunction. Routing dysfunction can be presented in different aspects. On one hand, *incorrect routing* involves attacks where a node routes messages incorrectly (or drops them) [23]. These attacks might not be relevant in P2P networks that do not provide multihop routing. Due to the flooding mechanism used to propagate information through the network, the consequences of a single node dropping messages are negligible.

On the other hand, in a *fake routing update*, the attacker tries to corrupt a given route (equivalent to corrupting a routing table for a given node) [23, 32, 48]. As we have mentioned previously in Section 4, there are no routing tables in the Bitcoin network. The most similar information a peer stores is addresses from other peers. Note that no information about where is this peer in the network nor its connections are stored by the Bitcoin client, just the address and a timestamp. Therefore, the attack that better resembles fake routing updates in Bitcoin is to send fake addresses. These kinds of attacks are usually performed as a first step in eclipse attacks, attacks already described in Section 5.2.

5.4.10. Tampering with Message Bodies. When using multihop routing, intermediate nodes can modify the content of the relaying packets. End-to-end integrity has to be provided in order to, at least, detect these types of attacks [16].

Tampering with the content of a block changes its hash and, with very high probability, invalidates its proof of work. Therefore, tampering with block data is not a feasible attack on Bitcoin. On the other hand, transactions are a signed data structure, with the signature cryptographically protecting its integrity. Therefore, an attacker can not tamper with a transaction to its will, for instance, by changing the destination address of the bitcoins transferred on the transaction. There is,

however, a very specific situation where this kind of attack would be possible (although the countermeasures that prevent this attack are already implemented and ready to be deployed). Because Bitcoin transactions are malleable, it is indeed possible for an attacker to change some part of the transaction while keeping the signature valid. This happens mainly because not all parts of the transaction are signed (e.g., the signatures themselves are not signed). The aforementioned situation where malleability is a problem for Bitcoin happens when a user is dealing with 0-confirmation transactions, that is, transactions that have been sent to the network but have not yet been included in a block. Because transactions are not yet in a block, an attacker may change some of the unsigned part of the transaction, creating another valid transaction that spends the same inputs but has a different identifier (recall that transactions are identified by their hash). Then, if this transaction is part of a protocol where transactions are identified by their hash, the attacker may be able to use it at his advantage.

6. Cryptocurrencies as Building Blocks for Decentralized Applications

Cryptocurrencies are indeed a powerful tool for the development of new decentralized applications (currently, the best well-known application for P2P networks is as a content distribution technology [49]), thanks to the distributed trust mechanism in which they are based on. Three relevant properties of cryptocurrencies can be used as a building block for such applications:

(i) Secure distributed payment mechanism

(ii) Distributed storage with integrity by design

(iii) Secure transfer and distribution of digital assets

The obvious use case of cryptocurrencies is, of course, to adopt them as the payment layer in any system where there is the need to transfer money from a payer to a payee in a totally distributed (and uncensored) fashion. Multiple applications could benefit from a flexible payment system, from P2P distributed storage schemes, where users could hire local disk space for an economical incentive, to more sophisticated scenarios, like mobile crowdsensing [50]. Mobile crowdsensing (MCS) is a distributed application where the power of the crowd, jointly with the sensing capabilities of smartphones they wear, provides a powerful tool for data sensing, especially in those scenarios involving user behavior or those that rely on user mobility, where standard sensor networks may not be suitable. However, including human participation in sensing tasks carries, at least, three critical challenges [51]: user participation, data sensing quality, and user anonymity. User participation is extremely important in MCS since the performance and usefulness of such sensor networks heavily depend on the crowd sensor's willingness to participate in the data collection process. Therefore, incentive mechanisms are of utmost importance in MCS scenarios to engage as many crowd sensors and provide the data collection center with a considerable wealth of data. User participation can be promoted by providing a pay-per-sense mechanism. However, standard payment schemes have multiple drawbacks in a pay-per-sense application. First of all, user enrollment in the payment system entails a burden step for user participation. Second, collateral costs of standard payment systems (mainly in the form of fees) prevent their use in a pay-per-sense scenario. Finally, standard payment mechanisms do not provide privacy-preserving properties, specially relevant when such payments could identify sensed data from a particular individual whose identity should not be disclosed. Cryptocurrencies can be successfully used in mobile crowdsensing scenarios as a rewarding mechanism since they allow an affordable pay-per-sense scheme with relevant privacy-preserving properties, as it has been proposed in the Paysense system [52].

Distributed storage is also a very interesting property offered by blockchain-based cryptocurrencies, but despite other distributed proposals, its main advantage is the integrity-by-design property that makes it so attractive for multiple applications. For instance, multiple P2P networks need a distributed IP/name resolution mechanism, and special purpose cryptocurrencies can solve this problem. An example of such an idea is the cryptocurrency Namecoin [53]. Namecoin is a blockchain-based cryptocurrency whose purpose is to provide network address resolution for network identifiers, normally human readable. In such a cryptocurrency, transactions can store data for tying the network address with other identifiers, and such transactions are stored in the blockchain inhering its integrity properties. Keys used for creating the transaction provide an authentication token for the owner who registered the tie. Namecoins could be used as a DNS replacement in a P2P network or even for node authentication when such authentication needs a tie between identity and keys (for instance, using standard public key infrastructure, PKI).

Cryptocurrencies were designed to transfer money, but its use can be extended to transfer other types of digital assets. By using a cryptocurrency as a transport layer, digital assets can be associated with cryptographic keys and can be traded, using the secure information included in the blockchain to determine the legitimate owner of every asset at each specific time. Multiple examples of such digital assets can be found, from shares of a company to DRM where the property of the media object can be determined [54]. Furthermore, extending a bit the concept of asset, cryptocurrencies can also be used to store reputation, conceptualized as an asset that users can store and transfer. Revisiting again the example of mobile crowdsensing, we recall the fact that data-sensing quality was one of the important challenges of such a scenario. In MCS systems, there is no control over the crowd sensors, and it cannot be assumed that all individuals will behave in the exact same manner or will be equally honest. Therefore, the overall quality of the sensor readings can see itself deteriorated if counterfeit data are received from malicious users. Hence, data validation techniques should be properly deployed, and a commonly used approach is to validate the data depending on the trust level of the crowd sensor that reports it. In this particular scenario, cryptocurrencies can be used as an annotation mechanism [52], by which users earn or lose reputation depending on the correctness of previous actions, accounted by the amount of rewards that they previously obtained.

Notice that, in this section, we have pointed out some possible uses of cryptocurrencies in distributed applications to show the broad intersection between both fields. However, an in-depth study on how interaction could be optimally performed between cryptocurrencies and particular scenarios is left for future work.

7. Conclusions and Further Research

In this paper, we have characterized P2P cryptocurrency networks by providing a deep analysis of the most relevant cryptocurrency nowadays: Bitcoin. By characterizing P2P cryptocurrency networks using well-known taxonomy in the field of P2P networks, we can conclude that such networks present a new paradigm due to the main properties that a cryptocurrency has to provide: reliability and security.

P2P cryptocurrency network reliability stands on top of a strong redundant mechanism regarding system information. As a result, every peer of the network stores all the relevant information of the system. With this approach, the availability of a single node in the network contains the information to keep all the systems alive. Notice that this approach turns out to a high inefficiency level regarding storage space, so this strategy is not followed by any other P2P network paradigm. Furthermore, such an approach also demands new synchronization mechanisms to provide all nodes with the same correct information.

Information redundancy is also used in the security plane for network topology protection. As we have seen, the main attacks to cryptocurrencies are eclipse attacks, where a victim or part of the network can be isolated. Such attacks can be performed when an attacker takes advantage of his position on the network topology. To avoid such possibility, the network topology has to be protected, and cryptocurrency networks use two different measures for such protection. On one hand, routing information should not be disclosed, so cryptocurrency networks are not multihop networks, and network nodes only are aware of one-hop neighbors. Using this approach, no routing information has to be provided to network nodes, and there is no restriction regarding information availability since, as we have pointed out above, information is replicated in every network node. On the other hand, network topology disclosure also has to be protected when nodes access the network. In such a phase, cryptocurrency P2P networks use a pseudorandom approach to determine each node connection to hinder the topology structure of the network. Notice that this topological secrecy property of P2P cryptocurrency networks is not so relevant in other P2P network paradigms, and for that reason, the mechanisms to achieve it are also particular of such environments.

Furthermore, some mechanisms specifically designed for other P2P network paradigms are not needed in cryptocurrency networks due to the characteristics of the information flowing in such networks. This is the case of multiple secure protections that try to prevent different attacks. For instance, intrinsic cryptographic properties of blocks and transactions can directly prevent DoS attacks, replay attacks, or tampering with message bodies.

As a new paradigm, P2P cryptocurrency networks open new research opportunities both as a direct field of study and also as a tool for other applications. For instance, a more formal analysis should be performed towards the pseudorandom mechanisms used in this kind of networks for selecting the nodes to connect to verify that network topology is both unknown and uniform. Furthermore, the development of a global P2P cryptocurrency network that could provide service to multiple cryptocurrencies, taking into account different particularities of each cryptocurrency, could also be another interesting research line. On the other hand, analyzing how such a new network paradigm could be efficiently combined and integrated with other distributed applications could also be a relevant topic for future work.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is partially supported by the Spanish Ministry under Grant no. TIN2014-55243-P and the Catalan AGAUR grant 2014SGR-691.

References

[1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008, https://bitcoin.org/bitcoin.pdf.

[2] A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies*, Princeton University Press, Princeton, NJ, USA, 2016.

[3] J. A. D. Donet and J. Herrera-Joancomartí, "Cryptocurrency P2P networks: a comparison analysis," in *Actas de la XIV Reunión Espanõla de Criptología y Seguridad de la Información (RECSI 2016)*, J. L. Ferrer and M. Payeras, Eds., pp. 423–428, Menorca, Illes Balears, Spain, 2016.

[4] M. Lischke and B. Fabian, "Analyzing the bitcoin network: the first four years," *Future Internet*, vol. 8, no. 1, p. 7, 2016.

[5] J. Herrera-Joancomartí and C. Pérez-Solà, "Privacy in bitcoin transactions: new challenges from blockchain scalability solutions," in *Modeling Decisions for Artificial Intelligence*, pp. 26–44, Springer, Berlin, Germany, 2016.

[6] A. Back, *Hashcash-A Denial of Service Counter-Measure*, 2002.

[7] Bitcoin Core, *Bitcoin Core Version 0.12.0 Released*, 2016, https://bitcoin.org/en/release/v0.12.0.

[8] Bitcoin Core, *Bitcoin Core Version 0.13.0 Released*, 2016, https://bitcoin.org/en/release/v0.13.0.

[9] Bitnodes: 2016, https://bitnodes.21.co/.

[10] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *IEEE Communications Surveys and Tutorials*, vol. 7, no. 2, pp. 72–93, 2005.

[11] P. Kirk, *Gnutella Protocol Development*, 2011.

[12] C. Decker and R. Wattenhofer, "Information propagation in the bitcoin network," in *Proceedings of the IEEE Internation Conference on Peer-to-Peer Computing (P2P)*, Trento, Italy, 2013.

[13] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proceedings of the 24th USENIX Conference on Secrity Symposium (SEC'15)*, pp. 129–144, USENIX Association, Berkeley, CA, USA, 2015, http://dl.acm.org/citation.cfm?id=2831143.2831152.

[14] D. S. Wallach, "A survey of peer-to-peer security issues," in *Software Security—Theories and Systems*, pp. 42–57, Springer, Berlin, Germany, 2003.

[15] S. Bellovin, "Security aspects of Napster and Gnutella," in *Proceedings of the 2001 Usenix Annual Technical Conference*, Boston, MA, USA, June 2001.

[16] D. S. Touceda, J. M. Sierra, A. Izquierdo, and H. Schulzrinne, "Survey of attacks and defenses on P2PSIP communications," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 3, pp. 750–783, 2012.

[17] Bitcoin Core, *Blockchain Info Website*, https://blockchain.info/.

[18] M. Carlsten, H. Kalodner, S. M. Weinberg, and A. Narayanan, "On the instability of bitcoin without the block reward," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, pp. 154–167, ACM, Vienna, Austria, 2016.

[19] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: routing attacks on cryptocurrencies," in *Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP)*, pp. 375–392, San Jose, CA, USA, 2017.

[20] K. Suto, H. Nishiyama, N. Kato, T. Nakachi, T. Fujii, and A. Takahara, "THUP: a P2P network robust to churn and DoS attack based on bimodal degree distribution," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 247–256, 2013.

[21] W. Wang, Y. Xiong, Q. Zhang, and S. Jamin, "Ripple-stream: safeguarding P2P streaming against DoS attacks," in *Proceedings of the 2006 IEEE International Conference on Multimedia and Expo*, pp. 1417–1420, Toronto, ON, Canada, July 2006.

[22] M. Brinkmeier, G. Schäfer, and T. Strufe, "Optimally DoS resistant P2P topologies for live multimedia streaming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 6, pp. 831–844, 2009.

[23] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *SIGOPS Operating Systems Review*, vol. 36, pp. 299–314, 2002.

[24] A. Singh, M. Castro, P. Druschel, and A. Rowstron, "Defending against eclipse attacks on overlay networks," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop (EW'11)*, ACM, Leuven, Belgium, 2004.

[25] N. Borisov and J. Waddle, "Anonymity in structured peer-to-peer networks," Tech. Rep. UCB/CSD-05–1390, Computer Science Division (EECS), University of California, Oakland, CA, USA, May 2005.

[26] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using P2P network traffic," in *Financial Cryptography and Data Security*, N. Christin and R. Safavi-Naini, Eds., vol. 8437 of Lecture Notes in Computer Science, pp. 469–485, Springer, Berlin, Germany, 2014.

[27] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed., vol. 7859 of Lecture Notes in Computer Science, pp. 6–24, Springer, Berlin, Germany, 2013.

[28] E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security*, A.-R. Sadeghi, Ed., vol. 7859 of Lecture Notes in Computer Science, pp. 34–51, Springer, Berlin, Germany, 2013.

[29] M. Ober, S. Katzenbeisser, and K. Hamacher, "Structure and anonymity of the bitcoin transaction graph," *Future Internet*, vol. 5, no. 2, pp. 237–250, 2013.

[30] A. Gervais, S. Capkun, G. O. Karame, and D. Gruber, "On the privacy provisions of Bloom filters in lightweight bitcoin clients," in *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC'14)*, pp. 326–335, ACM, New Orleans, LA, USA, December 2014.

[31] D. Cerri, A. Ghioni, S. Paraboschi, and S. Tiraboschi, "ID mapping attacks in P2P networks," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'05)*, vol. 3, pp. 1785–1790, St. Louis, MO, USA, 2005.

[32] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, Cambridge, MA, USA, vol. 2429 of Lecture Notes in Computer Science, pp. 261–269, Springer, Berlin, Germany, 2002.

[33] J. J. Douceur, "The sybil attack," in *Proceedings of 1st International Workshop on Peer-to-Peer Systems (IPTPS)*, Cambridge, MA, USA, March 2002.

[34] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: analysis and defenses," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04)*, pp. 259–268, Berkeley, CA, USA, April 2004.

[35] B. N. Levine, C. Shields, and N. B. Margolin, "A survey of solutions to the sybil attack," Tech. Rep. 2006-052, University of Massachusetts Amherst, Amherst, MA, USA, October 2006.

[36] J. Dinger and O. P. Waldhorst, "Decentralized bootstrapping of P2P systems: a practical view," in *Proceedings of the 8th International IFIP-TC 6 Networking Conference on NETWORKING 2009*, pp. 703–715, Springer, Aachen, Germany, May 2009.

[37] C. Cramer, K. Kutzner, and T. Fuhrmann, "Bootstrapping locality-aware P2P networks," in *Proceedings of the 12th IEEE International Conference on Networks, 2004 (ICON'04)*, vol. 1, pp. 357–361, Singapore, 2004.

[38] H. Tran, M. Hitchens, V. Varadharajan, and P. Watters, "A trust based access control framework for P2P file-sharing systems," in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, p. 302c, Washington, DC, USA, 2005.

[39] Y. Zhang, X. Li, J. Huai, and Y. Liu, "Access control in peer-to-peer collaborative systems," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCS'05)*, pp. 835–840, Columbus, OH, USA, June 2005.

[40] A. Ghodsi, L. O. Alima, and S. Haridi, "Symmetric replication for structured peer-to-peer systems," in *Proceedings of the International Workshops on Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P 2005/2006)*, pp. 74–85, Springer, Trondheim, Norway, August 2006.

[41] H. Weatherspoon and J. D. Kubiatowicz, "Erasure coding vs. replication: a quantitative comparison," in *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, pp. 328–337, Springer, Cambridge, MA, USA, March 2002.

[42] D. Hughes, G. Coulson, and J. Walkerdine, "Free riding on Gnutella revisited: the bell tolls?," *IEEE Distributed Systems Online*, vol. 6, no. 6, 2005.

[43] S. J. Nielson, S. A. Crosby, and D. S. Wallach, "A taxonomy of rational attacks," in *Proceedings of the 4th International Workshop on Peer-to-Peer Systems IV (IPTPS 2005)*, pp. 36–46, Springer, Ithaca, NY, USA, February 2005.

[44] M. Feldman and J. Chuang, "Overcoming free-riding behavior in peer-to-peer systems," *SIGecom Exchanges*, vol. 5, no. 4, pp. 41–50, 2005.

[45] C. Jennings, B. Lowekamp, E. Rescorla, S. Baset, and H. Schulzrinne, *Resource Location and Discovery (Reload) Base Protocol*, IETF, RFC 6940, January 2014.

[46] D. Goldschlag, M. Reed, and P. Syverson, "Onion routing for anonymous and private internet connections," *Communications of the ACM*, vol. 42, no. 2, pp. 39–41, 1999.

[47] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Financial Cryptography and Data Security, Lecture Notes in Computer Science*, N. Christin and R. Safavi-Naini, Eds., pp. 436–454, Springer, Berlin, Germany, 2014.

[48] S. Marti, P. Ganesan, and H. Garcia-Molina, "SPROUT: P2P routing with social networks," in *Proceedings of the Current Trends in Database Technology EDBT 2004 Workshops: EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb*, pp. 425–435, Springer, Heraklion, Crete, Greece, March 2004.

[49] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Computing Surveys*, vol. 36, no. 4, pp. 335–371, 2004.

[50] W. Zamora, C. T. Calafate, J.-C. Cano, and P. Manzoni, "A survey on smartphone-based crowdsensing solutions," *Mobile Information Systems*, vol. 2016, Article ID 9681842, 26 pages, 2016.

[51] B. Guo, Z. Wang, Z. Yu et al., "Mobile crowd sensing and computing: the review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, vol. 48, no. 1, pp. 1–31, 2015.

[52] S. Delgado-Segura, C. Tanas, and J. Herrera-Joancomartí, "Reputation and reward: two sides of the same bitcoin," *Sensors*, vol. 16, no. 6, p. 776, 2016.

[53] H. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of Namecoin and lessons for decentralized namespace design," in *Proceedings of the 14th Workshop on the Economics of Information Security (WEIS'15)*, Delft, Netherlands, June 2015, http://randomwalker.info/publications/namespaces.pdf.

[54] M. Kitahara, J. Kawamoto, and K. Sakurai, "A method of digital rights management based on bitcoin protocol," in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication (ICUIMC'14)*, pp. 84:1–84:6, ACM, New York, NY, USA, 2014.