Bitcoin and Blockchain: Security and Privacy

Ehab Zaghloul, Tongtong Li, *Senior Member, IEEE*, Matt W. Mutka, *Fellow, IEEE*, and Jian Ren, *Senior Member, IEEE*

***Abstract*—Blockchain is a technology that was proposed to enable the decentralized digital currency, Bitcoin. Since its inception, blockchain has been widely used in many other areas, including tracing sensor data and mitigating its duplication in IoT applications, the healthcare industry, and e-voting. In this article, we provide a comprehensive review and analysis of the major security and privacy issues of Bitcoin and blockchain, the major challenges, and opportunities in utilizing the technology. First, we present a comprehensive background of Bitcoin and the preliminary on security. Second, the major security threats and countermeasures of Bitcoin are investigated. We analyze the risk of double-spending attacks, evaluate the probability of success in performing the attacks, and derive the profitability for the attacker to perform such attacks. Third, we analyze the underlying Bitcoin peer-to-peer network security risks and Bitcoin storage security. We compare three types of Bitcoin wallets in terms of security, types of services, and their tradeoffs. Finally, we discuss the security and privacy features of alternative cryptocurrencies and present an overview of emerging technologies today. Our results can help Bitcoin users to determine a tradeoff between the risk of double-spending attempts and the transaction time delay or confidence before accepting transactions. These results can also assist miners to develop suitable strategies to get involved in the mining process and maximize their profits.**

***Index Terms*—Bitcoin, blockchain, cryptocurrency, digital assets, double-spending.**

I. Introduction

A CRYPTOCURRENCY is a decentralized online currency that was developed as an alternate means to transfer money in an unprecedented way. The existing financial systems require a centralized trusted financial institution to securely process transactions between two parties. This institution charges costly service fees that are unavoidable for banking customers. In addition to such cost burdens, delayed processing time and security issues have affected the modern-day financial industry. Certain transactions, such as funds transfer, may take days or weeks to be cleared, causing issues in cases of urgency. The modern-day financial system is also plagued with security and privacy vulnerabilities. Financial institutions employ the most advanced security techniques to protect customers. However, the sensitive information of the customer is always exposed to the financial institutions making it vulnerable to information leakage. To mitigate these security concerns, privacy risks, and inconveniences, new cryptographic protocols have been developed to allow secure and convenient asset transfer, without involving a centralized third party.

In 2008, Nakamoto [1] developed a white paper in which he proposed Bitcoin. Bitcoin is an online peer-to-peer (P2P) digital cash system that does not require a trusted third party. In Bitcoin, users possess ownership rights to virtual cryptocoins that are denoted as Bitcoins (BTCs). Users generate transactions to transfer BTC and store them in the public ledger, blockchain. The smallest transferable value today is known as a Satoshi, which is equivalent to one-hundredth of a millionth BTC (i.e., 0.00000001 BTC).

Bitcoin transactions utilize cryptographic protocols to provide a secure process while striving to preserve the privacy of both the buyer and seller. The transactions are stored in a blockchain [2]–[5] to limit inherent issues of digital media such as double-spending [6]. A blockchain is a distributed database acting as a public ledger that holds all processed transactions. It is based on a distributed consensus that allows any past and present online transaction to be verified [7].

Bitcoin transactions are released into the network and validated by the nodes as they propagate through the entire network. The validating nodes referred to as *miners*, compete to mine groups of transactions into blocks and earn BTC as a reward. Mining is the process of solving a hard cryptopuzzle, referred to as the Proof-of-Work (PoW), that requires extensive computational power. The first miner capable of finding a solution to the problem broadcasts his/her block to the network and earns the reward. The reward consists of a specified amount of newly released BTC and all transaction fees associated with the transactions included in the block. All other miners then surrender to the solution of the winning miner and append the winning block to the blockchain.

The first Bitcoin software was implemented by Satoshi Nakamoto and is known as the Bitcoin core. This implementation is sometimes referred to as the *Satoshi client* and is run by most of the network nodes in Bitcoin. It is an open-source project with a large developer community contributing to it. The developers follow a bitcoin improvement proposals (BIPs) [8] document and introduce the standards of the system. The document also contains new features and proposals for the developer community to test thoroughly before making final modifications to the software.

Following Bitcoin, many cryptocurrency systems appeared and continue to do so today. The blockchain technology is a common characteristic shared by many newly emerging cryptocurrency systems [9]. The majority of these systems are

Manuscript received May 7, 2020; accepted June 14, 2020. Date of publication June 22, 2020; date of current version October 9, 2020. This work was supported in part by NSF under Grant CCF-1919154 and Grant ECCS-1923409. *(Corresponding author: Jian Ren.)*

The authors are with the Department of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824 USA (e-mail: ebz@msu.edu; tongli@msu.edu; mutka@msu.edu; renjian@msu.edu).

Digital Object Identifier 10.1109/JIOT.2020.3004273

mainly clones of Bitcoin. These systems introduced only minor adjustments such as currency supply or block size within the Blockchain. Alternatively, a few systems introduce innovative concepts that offer substantial features. Examples of these features include novel consensus mechanisms or enhanced decentralized computing platforms that can provide additional functions and higher flexibility to the system.

All cryptocurrencies are traded in the online cryptocurrency marketplace. The cryptocurrency market is similar to other exchange markets such as the stock market, with various trading platforms. However, the cryptocurrency market is not regulated by a government or agency and trading occurs virtually 24/7 across the world. The nature of cryptocurrency allows transactions to occur at speeds that cannot be accomplished with fiat currency, such as the United States dollar. This results in a much more volatile market than traditional trading markets. Coin prices are continuously rising and dropping, and new cryptocurrencies consistently enter and leave the marketplace. Many coins continue to rise in value based on value demonstrated to investors. However, increased speculation in the marketplace has lead to the overevaluation of many cryptocurrencies.

As of January 2020, the total market cap of the cryptocurrency market hit $211 billion [10]. This amount comes from the total valuation of almost 1000 cryptocurrencies on the market today. Comparing this amount to the $17.6 billion total market cap in 2016, the market has increased by 1198%. This rapid growth in the new market has led an effort to examine the role of cryptocurrency in the future.

A. Contributions

While the main purpose of this article is to examine the potential stability of Bitcoin, we strive to delve deeper into analyzing the double-spending attacks by modeling the probability of success in multiple ways. In particular, utilizing our analysis, we present our own profitability analysis of the double-spending attacks. We reveal a breakpoint in time when attackers should give up on the attack since it is unlikely that they will turn a profit beyond this point (i.e., the time when the cost is greater than the revenue). We present a tradeoff between the waiting time before accepting a transaction versus the profits/losses of the attackers. This may help maximize the confidence of the users before accepting transactions. In addition to this, we thoroughly analyze the infrastructure of Bitcoin storage wallets. Our discussion presents the key algorithmic features introduced by each wallet type in order to counter the different potential threats. The main purpose is to enlighten the users with the tradeoffs when using different types of wallets from a cryptographic perspective.

More specifically, the major contributions of this article can be summarized as follows.

1) We provide a comprehensive explanation of the primary components of Bitcoin discussed in sequential and logical order for the readers to comprehend. The main purpose is to cultivate the readers with the necessary background on Bitcoin to consolidate their understanding of the system.
2) We delve thoroughly into the analysis of double-spending attacks. We first show that the probability of success of performing double-spending attacks can be modeled using two distinct probabilistic models. We show that both models result in a similar outcome. Next, using these probabilistic models, we present a profitability analysis on performing double-spending attacks. The main purpose of this analysis is to reflect the tradeoff between the waiting time before accepting a transaction versus the profits/losses of the attackers. We also aim at reflecting that attackers with 51% computational power or more will continue to profit indefinitely.
3) We present fundamental network security and privacy concerns. The purpose of this analysis is to expand the knowledge of readers on major security and privacy concerns that threaten the stability of systems running the blockchain technology. Our target is to help the reader realize the major threats to such systems from a security and privacy perspective.
4) We dive deeply into the exploration of Bitcoin storage wallets. We first classify wallets based on their underlying infrastructure and methods of PKI pair generation. We aim at presenting the cryptographic primitives related to each type of wallet. Next, we classify wallets based on installation environments and then further classify them based on functionality. We strive to help the readers understand the different classes of wallets, their corresponding security risks, and the best practices to secure their cryptocoins.

The interest in blockchain continues to grow aggressively. It has already attracted a wide range of audiences, such as governments, enterprises, healthcare, and many more. We realize that in order for the blockchain to sustain its success and for these interested entities to adopt it, we must educate a wider range of audience, which could include: 1) researchers at the beginning of the line who wish to expand on research in this area and 2) skeptical entities and individuals who wish to adopt the technology and wish to learn more about it. We aim at putting together a comprehensive study that explores the blockchain technology from multiple angles and filling in the gaps of previous studies.

B. Organization

The remainder of the article is organized as follows. In Section II, we provide a comprehensive background review on Bitcoin outlining its building blocks and protocols. Next, in Section III, we evaluate double-spending attacks and present our profitability analysis. Following that, in Section IV, we assess the major network security attacks of the Bitcoin network. In Section V, we analyze the security issues in the storage wallets used by Bitcoin today. We investigate the subsequent privacy protocols of Bitcoin in an effort to limit the linkage problem in Section VI. In Section VII, we review protocols and alternative consensus algorithms implemented in emerging cryptocurrencies outlining the security and privacy advantages and limitations. Finally, we conclude our study,

summarize the lessons learned, and future research directions in Section IX.

II. Understanding Bitcoin

Research in digital cash dates back to the early 1980s [11]. In 1990, DigiCash Inc., an electronic cash corporation, made an initial attempt to provide a cryptocurrency system [12]. Later, other systems, such as DigiCash [12] and b-money [13] were introduced that used cryptographic protocols and aimed to enhance privacy. These systems suffered key issues including double-spending attacks and did not use the blockchain technology.

Blockchain as a concept was initially proposed by Haber and Stornetta in 1991 [2]. Their blockchain aimed at certifying the creation/modification of a digital record by digitally timestamping the records. However, the blockchain was not efficient since each record was independently timestamped. To improve efficiency, Merkle trees [14] were incorporated into blockchains in 1992 [3]. They improved efficiency by handling multiple digital records into one block. Finally, Nakamoto implemented the first real blockchain and used it as the core technology for the Bitcoin cryptocurrency system. In this section, we will present the major building blocks and protocols of Bitcoin.

A. Bitcoin Network

Bitcoin runs over a P2P network. The main advantage of using a P2P network is the agile movement of data for all nodes to achieve consensus. In contrast to the typical P2P network used to share data files between interested peers, Bitcoin utilizes the network to rapidly broadcast data among all the connected nodes. This process is known as *flooding* and continues until all nodes within the network receive the broadcast data.

It is important to differentiate between the terms *node* and *peer* of a P2P network. A node is a network entity that is connected to one or multiple other similar nodes. The directly connected nodes are referred to as peers. Nodes propagate data to the indirectly connected nodes by traversing it to their peers which follow a similar manner until the data reaches every connected node.

In the Bitcoin network, data being flooded include IP addresses of the nodes, newly generated transactions, and blocks of verified transactions that extend the blockchain. Peers share IP addresses of other nodes that they are connected to or have discovered from their peer nodes. The aim behind sharing IP addresses is to allow peers in the network to discover and connect to more nodes resulting in a random network topology. Newly generated transactions are broadcast through the network to rapidly publicize their occurrence to all connected nodes. Miners compete to mine these transactions into blocks. The winning miner broadcasts the block to all the nodes to extend and update their version of the blockchain.

Nodes in the Bitcoin P2P network are defined based on their roles. The main duties are summarized as transaction generation, block/transaction routing, block/transaction verification, and transaction mining. Block/transaction routing is performed by all nodes.

A node that can perform all functions is referred to as a *full node*. It consistently keeps a copy of the full blockchain allowing it to verify any transaction without needing the assistance of other connected nodes. It also possesses a BTC wallet that can generate transactions and compute the possessed value of BTC by the node. Moreover, a full node possesses computational resources to compete in the mining competition. Nodes that do not store a full copy of the blockchain are referred to as simplified payment verification (SPV) nodes or *lightweight nodes*. These nodes require assistance from full nodes when verifying a transaction. Full nodes feed the SPV nodes with the required information from the blockchain necessary to complete the transaction verification.

Some nodes may only perform one particular function. Ones that are engaged in the mining process are referred to as *mining nodes* while others that generate transactions are referred to as *wallets*.

In most Bitcoin software implementations, all nodes are treated equally and can be uniquely identified by their IP addresses. Using these addresses, peers establish transmission control protocol (TCP) connections with one another. Each node can choose whether to connect to the network using a public or private IP. A node that utilizes a public IP is accessible over the Internet by any connected node while one with a private IP is only accessible by nodes within its private network. By default, a node with a public IP address is granted eight *outbound* connections and 117 *inbound* connections, resulting in a total of 125 connections. On the other hand, a node with a private IP address is granted only eight *outbound* connections. An outbound connection is initiated by the node itself when it requests connecting to a discoverable node while an inbound connection is initiated by other nodes in the network that desire connecting to the node.

For explanation purposes, we define the node that initiates a connection as *client* and the node that waits for an incoming connection as *server*. Both nodes engage in a TCP handshake by exchanging network packets defined as version and verack. The client initiates a connection request by sending a version packet addressed to the IP address of the server. By default, the server listens on port 8333 for incoming version packets. If the server accepts the version packet, it responds with a verack packet and its own version packet, both addressed to the IP address of the client. Finally, the client responds by sending a verack packet addressed to the IP address of the server and the connection is established. The connection enables symmetric communication allowing the client and server to exchange data bidirectionally. The connection is lost if peers do not communicate for a specified idle time. To reconnect, peers engage in a new TCP handshake.

As discussed previously, a node shares with its peers a list of IP addresses that it has learned as a result of being connected to the network. Each node stores its list in two separate tables: 1) a *tried* table and 2) a *new* table. The tried table of a node stores IP addresses that the node has established connections with while its new table stores IP addresses that it has only discovered but did not attempt to connect to yet. When a node

desires sharing IP addresses with its peers, it randomly selects IP addresses from both tables and sends them in addr messages. An addr message can contain up to 23% or a maximum of 1000 IP addresses of the total IP addresses stored in both tables. To initiate sharing, a node sends a getaddr message to its peers requesting them to share their lists of IP addresses. The peers then respond with an addr message. In some cases, sharing IP addresses is unsolicited if a node voluntarily sends an addr messages to its peers without receiving a getaddr message.

A node that wishes to connect to the Bitcoin network for the first time cannot obtain IP addresses by this method. Bootstrapping is mainly achieved by communicating with a domain name server (DNS) seeder. The node sends a DNS query requesting a list of active IP addresses. If the DNS fails to respond with an appropriate list of active IP addresses, the node can still connect to the network by using a hard-coded list of IP addresses, referred to as *seeds*. Once connected to any of these IP addresses, the node can then request more IP addresses from its peers by sending getaddr messages.

Nodes also relay verified transactions and blocks to their peers to reach consensus. A node begins by broadcasting an inventory (inv) message to all its peers informing them of the new transactions or blocks it has received and verified. The peer nodes check whether they are already informed of these new transactions and blocks then respond to the node with a getdata message. The getdata message includes all transactions and blocks a peer node is not aware of. The node then responds with a transaction/block message that includes the complete transactions/blocks the peer requests. Once received, the peer validates the transactions or blocks and continues to relay them to its own peers in a similar manner. If a received transaction or block cannot be validated, it is immediately dropped and its propagation is discontinued.

B. Bitcoin Transactions

We define a Bitcoin transaction (TX) as the transfer of an amount of BTC *ownership rights* from the wallet of the buyer to the wallet of the seller, in exchange for a product or service. BTC wallets utilize elliptic curve digital signatures to handle the transfer of ownership rights and ensure that unauthorized spending of the cryptocurrency is infeasible. Each wallet randomly generates a private key Pr that is used to derive its corresponding public key Pub that is shared among all users. The Pub is used to generate the address of the wallet needed to make payments to it while Pr is used to generate a digital signature corresponding to Pub in order to claim payments made to the wallet and use them in later transactions. A Pr is first generated from a cryptographically secure pseudorandom number generator (CSPRNG) and its corresponding Pub is then calculated using the elliptic curve digital signature algorithm (ECDSA). Calculations are performed based on the field and curve parameters defined by secp256k1 with the curve order n [15] as follows:

$$\mathsf{Pr} = \mathsf{CSPRNG}(), \quad \mathsf{Pub} = \mathsf{Pr} \times G \pmod{n} \tag{1}$$

where G is a generator of the elliptic curve and $\times$ represents elliptic curve multiplication.

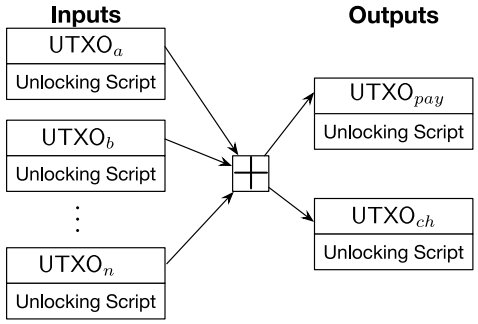

Fig. 1. Single transaction with multiple UTXO inputs and outputs.

The BTC wallet of the buyer assembles a transaction using the unspent transaction outputs (UTXO) of the buyer stored in the blockchain. An UTXO specifies an amount of BTC claimed earlier by the buyer as a result of a previously processed transaction. A simple BTC transaction is shown in Fig. 1.

In the figure, we show that a transaction can consist of multiple inputs and outputs. The output $\mathsf{UTXO}_{\text{pay}}$ represents the transfer of ownership rights of a certain amount of BTC from the wallet of the buyer to the wallet of the seller. The output UTXO_{ch} represents redirecting ownership rights of the BTC change amount back to the wallet of the buyer. A distinct *locking script* is attached to each of these outputs, which specifies conditions that must be met in order to grant ownership rights. For example, the locking script attached to $\mathsf{UTXO}_{\text{pay}}$ must include the Pub of the seller needed to generate his/her wallet address. This ensures that the payment is made to the wallet of the seller and only he/she is granted access to it with his/her corresponding Pr. Using Pr, the seller can generate a digital signature that corresponds to the Pub associated with the locking script, hence claim the output.

The inputs $\{\mathsf{UTXO}_1, \mathsf{UTXO}_2, \ldots, \mathsf{UTXO}_n\}$ represent unspent transaction outputs claimed by the buyer from the previous transactions. When a buyer decides to use a specific output from a previous transaction as an input to a new transaction, the buyer must specify proof that he/she still possesses ownership rights and did not previously spend them in another transaction. This is done by attaching an *unlocking script* to each input. The unlocking script solves the locking script that was associated with the output from the previous transaction. Likewise, the unlocking script is a digital signature produced by the Pr of the buyer that corresponds to a Pub associated with the locking script of an UTXO. A valid unlocking script is legitimate proof of continuous possession of ownership rights to certain BTC being used as input. As a result, BTC can be viewed as a chain of digitally signed transactions where ownership rights are transferred from one owner to the other by digitally signing them.

A transaction must include at least one input, however, may include multiple outputs to simultaneously pay different sellers from the total value associated with the inputs. The locking script of each output would specify the conditions of its claimer. However, it is necessary that the total BTC value of

the inputs is always equal to or greater than the total value of the outputs. In the event that the total value of the inputs is greater than the total outputs, the difference, known as the *transaction fee*, is rewarded to the miner that adds the transaction into a block attached to the blockchain. For guaranteed processing, most available wallets today derive the transaction fee as a fixed amount of BTC in relation to the size of the transaction. In other words, the transaction fee increases with the size of the transaction.

The wallet of the user combines all transaction inputs/outputs and their corresponding scripts into one digital message M. It then applies the secure hash algorithm SHA256 to M twice to increase security before releasing it into the network. The 32-B digest representing the identity of the transaction ($\mathsf{ID_{TX}}$) is generated as follows:

$$\mathsf{ID_{TX}} = \mathsf{SHA256}\big(\mathsf{SHA256}(\mathsf{M})\big). \tag{2}$$

A newly generated transaction assembled by the BTC wallet of a buyer is released into the Bitcoin network to be validated and stored in the blockchain. The generating node transfers the transaction to its peers that flood it to the rest of the network nodes. Each node that receives it, audits the inputs by executing the scripts associated with it. This audit involves checking whether the execution of the unlocking script integrated by a buyer within each input matches its corresponding locking script defined in the previous transaction. If a match exists, the node relays the transaction to its peers and temporarily places it in its *transaction pool* until chosen to be mined, otherwise, the transaction is dropped.

In some cases, transactions are not flooded into the network in the same order they are generated. As a result, during the audit, a node might not be aware of some inputs of a transaction (child transaction) referring to the outputs of other transactions (parent transactions). Instead of immediately rejecting the transaction and considering its inputs as invalid, the node can temporarily place it into an *orphan transaction pool*. If the parent transaction shows up, the inputs of the child transaction become valid and it can be transferred to the transaction pool.

C. Bitcoin Transaction Standards

There are five major Bitcoin transaction standards and a few nonstandard transactions. All transaction types are generated with a stack-based scripting language that is processed from left to right. A script consists of a list of instructions that must be executed in the correct order to grant an individual the right to spend the BTC within a transaction. The list of standards is described as follows.

Pay to Public Key Hash (P2PKH): This standard transaction is the most used type. The locking script within each output of a transaction holds the public-key hash (serving as a Bitcoin address) of the seller that will claim the BTC amount included. In other words, the locking script defines a condition that the seller must possess a specific Pr corresponding to the public-key hash to claim the output. Once claimed by the seller, the output becomes an UTXO owned by the seller. In order for the seller to use this specific UTXO as an input to a future transaction, the seller must attach a valid unlocking script to it. The unlocking script includes the Pub of the seller and a digital signature generated by his/her Pr that corresponds to the public-key hash associated with the locking script of the previous transaction output.

Pay to Public Key: The intent behind this standard transaction is to simplify the P2PKH standard. Rather than associating the public-key hash within the locking script of the output, the public key itself is used. As a result, the validation process is simple. The digital signature of the seller generated with a Pr can immediately be compared to the associated Pub by searching whether or not they match.

Multisignature (MultiSig): In this standard transaction, a combination of keys is required to authorize an output claim. The locking script of a transaction output is associated with a number (N) of public keys. In order for an individual to claim the output, the individual must possess M-of-N private keys that correspond to the N public keys. This type of transaction can increase security and can be used in scenarios that require more than one user to be present in order to claim and spend BTC. However, as the number N of public keys associated with the transaction output increases, the size of the transaction also increases. As a result, these transactions acquire large space in the UTXO pool, therefore, requiring more storage memory. As discussed previously, larger transactions also require larger transaction fees.

Pay to Script Hash (P2SH): This standard transaction was introduced to resolve the complex issues caused by MultiSig transactions. The transaction has the same simple complexity as a P2PKH transaction. Rather than associating the entire locking script with a transaction output that includes multiple public keys, a double hash computation is applied to the entire script, SHA256(RIPEMD160(*script*)). The result is a 20-B digest that is attached to the locking script instead of the entire original script. In order to use the output from this transaction as an input to another transaction, the buyer creates an unlocking script that holds M-of-N private keys and the original script that was cryptographically hashed earlier. In that way, sufficient information is available in the locking and unlocking scripts to validate the UTXO for spending. In addition, the buyer no longer has to worry about generating large transactions which might require hefty transaction fees to process. Instead, only the seller is required to provide the unlocking script he/she wishes to spend the output in a new transaction.

Data Output: This standard transaction is intended to store arbitrary data on the blockchain rather than transfer BTC from a buyer to a seller. In the Bitcoin community, many members believe that such transactions are abusive to the system since it places a burden on the network nodes to process transactions that do not carry BTC. However, such transactions exist and allow 40 B of data to be stored per transaction. These transactions are unspendable, therefore, are not stored in the UTXO set.

Nonstandard: A very small percentage of transactions are processed under nonstandard transactions. Nonstandard transactions use more sophisticated scripts that strive to provide higher complexity and security. In some cases, these

transactions might even be the result of bugs or mistakes resulting in loss of BTC.

D. Merkle Trees

Validated transactions are grouped into blocks that are then mined and stored in the blockchain. A single block can contain multiple transactions up to the block size limit. Merkle trees sometimes referred to as hash trees, are utilized to cluster multiple transactions in one block. A Merkle tree is a tree data structure generated in a bottom-up approach that can efficiently summarize and verify the integrity of the transactions being combined. Starting from the leaf nodes which are hashes of the original data, each nonleaf node is generated as a computation of its respective children nodes. For a single nonleaf node, all its children nodes are concatenated then hashed to produce a single digest that represents the node in the tree. The approach continues until a single node is generated which is defined as the root node.

BTC utilizes a binary Merkle tree in which each nonleaf node has exactly two children. It applies a double hash computation $\mathsf{SHA256}(\mathsf{SHA256}(\cdot))$ when generating nodes. The leaf nodes used to construct the tree are the identities $\mathsf{ID_{TX}}$ generated for each transaction as discussed in (2).

In a binary Merkle tree, each row within the tree consists of an even number of nodes, except the root node. In the case where a row consists of an odd number of nodes, a replica of the last node is reproduced to even out the number of nodes in that row. To better comprehend the construction of the binary Merkle tree, consider a block that consists of five transactions, $\{\mathsf{TX}_1, \mathsf{TX}_2, \ldots, \mathsf{TX}_5\}$. Each one of these transactions has already been validated by the nodes and identity for each transaction has been generated as discussed in (2). We denote the corresponding identities as $\{\mathsf{ID}_{\mathsf{TX}_1}, \mathsf{ID}_{\mathsf{TX}_2}, \ldots, \mathsf{ID}_{\mathsf{TX}_5}\}$, where each identity represents a leaf node in the tree. In this example, the number of nodes at the leaf node level is odd, therefore, a replica of the fifth identity is generated, $\{\mathsf{ID}_{\mathsf{TX}_1}, \mathsf{ID}_{\mathsf{TX}_2}, \ldots, \mathsf{ID}_{\mathsf{TX}_5}, \mathsf{ID}_{\mathsf{TX}_6} = \mathsf{ID}_{\mathsf{TX}_5}\}$. Next, the double hash computation is applied to the concatenation of each two identities to generate the parent nonleaf nodes of the Merkle tree as follows:

$$N_i = \mathsf{SHA256}\big(\mathsf{SHA256}\big(\mathsf{ID}_{\mathsf{TX}_{2i-1}} \| \mathsf{ID}_{\mathsf{TX}_{2i}}\big)\big), \quad i = 1, 2, 3 \tag{3}$$

where $\|$ is the concatenation of two identities.

As shown in the previous equations, an odd number of nonleaf nodes is generated at that level. To even it out, we replicate N_3 to produce N_4 as

$$N_4 = \mathsf{SHA256}\big(\mathsf{SHA256}\big(\mathsf{ID}_{\mathsf{TX}_5} \| \mathsf{ID}_{\mathsf{TX}_6}\big)\big). \tag{4}$$

Using the resulting digests, we can generate the following level of nonleaf nodes as:

$$N_{4+i} = \mathsf{SHA256}\big(\mathsf{SHA256}(N_{2i-1} \| N_{2i})\big), \quad i = 1, 2. \tag{5}$$

Finally, the 32-B root node is derived as

$$R = \mathsf{SHA256}\big(\mathsf{SHA256}(N_5 \| N_6)\big). \tag{6}$$

Fig. 2 represents the complete construction of the Merkle tree for this example. The dotted nodes represent the replicated nodes that are added to even out the odd rows. The root node, R, representing the summary of all transactions are placed into the block header of a block to be mined and chained to the blockchain.

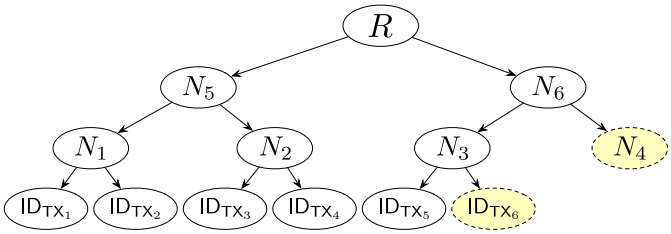

Fig. 2. Merkle tree within a block.

The use of Merkle trees is more common in SPV nodes since they do not store a copy of the full blockchain. When an SPV node needs proof of the existence of a transaction within a block, it turns to a full node for assistance. The full node will generate a *Merkle path* by computing a maximum of $\log_2 N$ $\mathsf{SHA256}(\mathsf{SHA256}(\cdot))$ computations, where N represents the total number of transactions in the tree. Using the Merkle path as an authentication path, the SPV node can prove the existence of a transaction within the tree. This proof of existence method is considered to be efficient since it only requires hash computations.

E. Blockchain

The blockchain is a public ledger that stores all previous transactions since the creation of Bitcoin. It provides its users with transaction confirmations to track ownership rights of BTC. As new transactions are processed, the blockchain is extended. It consists of blocks $\{\mathsf{B}_0, \mathsf{B}_1, \ldots, \mathsf{B}_n\}$, each carrying a set of validated transactions, where B_0 represents the first block and B_n represents the most recent block attached to the blockchain. Blocks are linked back-to-back, with each one referencing its previous block to form the complete blockchain. To reference a block, a unique 32-B identity $\mathsf{ID}_{\mathsf{B}_i}$ is generated for B_i by applying to the block header $\mathsf{SHA256}(\mathsf{SHA256}(\cdot))$. An identity is referred to as the block hash.

The head of the blockchain is denoted as B_0 and is defined as the *genesis block*. B_0 differs from all other blocks as it does not reference any previous block. At the launching stage of the system, B_0 was a stand-alone block waiting for the system to initiate a newly mined block to be chained to it.

Each block consists of two parts, a header, and a body. Each header incorporates the block hash of its predecessor block in the chain. The header also consists of a difficulty target, nonce, and a timestamp, which are discussed in more detail in the following section. The body carries all leaf nodes and nonleaf nodes of the Merkle tree, excluding the Merkle root, which is incorporated in the header. This design makes it infeasible to retroactively alter records within any block of the blockchain. Any modification to one block will require adjusting all the subsequent chained blocks.

F. Bitcoin Mining

Bitcoin mining is the final stage to secure validated transactions and add them to the blockchain. Once a transaction is added to the blockchain, it becomes completely verified and public to all users. The transaction claimer(s) can use the embedded UTXO(s) as the input to other transactions whenever desired.

Miners begin by selecting transactions from their transaction pools that will be placed into a block where a block cannot exceed 1 MB in size. A small portion of that space is specified to carry high-priority transactions. Priority is based on the size and age of the transaction inputs. The rest of the block is filled with other transactions that have greater transaction fees to maximize the profit that a miner can turn if successful in mining the block first. A transaction with low or no fees will probably remain in the transaction pool of the miner until it ages and becomes a high-priority transaction.

Next, the miner assembles a special transaction, known as the *coinbase transaction*. This transaction is a reward paying transaction to the miner in the event of winning a mining competition. It does not have any inputs and consists of a single output addressed to the wallet of the miner. The amount incorporated in the output is the reward mining fee (6.25 BTC at the time of writing) plus the sum of all transaction fees included in each transaction.

All the selected transactions along with the coinbase transaction are then combined into a Merkle tree as discussed previously. At this point, the miner has all components needed to construct the block header of the new block except the *nonce*. Miners compete in the PoW cryptopuzzle to derive the value of this nonce. This value, if concatenated with the block header of the group of chosen transactions and then double hashed, must produce a digest with a specific prefix of zeros in its binary representation. Searching for this value is performed in a brute-force manner and is directly correlated with the computational power available. The more computational power available, the faster a miner is able to find the correct nonce. A successful miner will then broadcast his/her PoW to claim the reward.

The primary advantage of the PoW cryptopuzzle is to make it computationally infeasible to perform Sybil attacks [16]. PoW is intentionally designed to be resource intensive to perform while efficient to verify. It requires a certain number of zeros to appear in the prefix of the digest as a result of applying the double SHA256 computation. The prefix determines the *difficulty* of finding the correct nonce. The more zeros required in the prefix, the harder it is to find the correct nonce and *vice versa*. The difficulty is dynamically altered every two weeks based on the number of miners (total hash power) so that the average time it takes for a miner to find a correct solution is approximately 10 min, a platform design choice.

The first miner to find the correct nonce to a block of transactions is rewarded a mining reward as compensation for the computational power spent. The mining reward is halved precisely every 210 000 blocks that are added to the blockchain. It is estimated to continue until the year 2140 when nearly 21 million BTC will have been released into the system. The reason for having a fixed supply of BTC is to prevent price inflation in the future.

Another incentive that encourages miners to spend their computational power to perform mining is the transaction fee. The winner is not only rewarded the mining reward but is also given all the transaction fees incorporated with all the transactions in the block. With time, the mining reward will decrease due to halving, which will demand higher transaction fees in the future to compensate for the reduced mining reward.

After a block is successfully mined, all miners check their transaction pools to eliminate the transactions that have been included in the mined block and immediately construct a new block of transactions. The end of the mining race marks the beginning of a new one. Miners instantly begin to search for the nonce of the next block of transactions.

Simultaneously, the mined block is flooded through the network so that all nodes can update their blockchains. The winning miner transmits the block to its peer nodes to validate it before propagating it further through the network. The peer nodes check whether the block is correctly assembled in terms of syntax and variables. The PoW provided by the miner must be correct and the coinbase transaction must include the correct amount to pay the miner. If any information is invalid, the block is immediately dropped.

Quite regularly, as blocks are mined to extend the blockchain, a temporary incident, known as a *fork*, might occur. A fork occurs when two miners are able to simultaneously mine two different blocks at the same time. As a result, both newly mined blocks are accepted to extend the blockchain. The blocks are flooded into the network and the miners update their version of the blockchain based on the block they receive first. This results in two valid versions of the blockchain in possession by the miners with two different paths. However, the miners continue to extend their version of the blockchain regardless which path they possess. Eventually, one path will grow longer than the other as mining continues. The path that grows longer is the winner and all nodes immediately discard the other path and update their blockchain to the longer one. In literature, the blocks that are dropped are known as *orphan blocks*; valid blocks that were part of the blockchain at some point.

G. Bitcoin Scalability

Scalability is a major performance issue of the blockchain systems. Layer 2 protocols, also known as off-chain protocols, have been proposed in efforts to improve these issues. Lightning network [17] is the most notable example. It operates atop the Bitcoin system to provide transaction speedup. It also runs over a P2P network allowing the nodes to perform Bitcoin micropayments. Users of the protocol initially open payment channels with designated BTC amounts. The connected nodes can then begin to perform micropayments among each other that are not settled on the blockchain but stored locally. Once a user decides to close a payment channel, the final balance of the channel is settled on Bitcoin's blockchain allocating the correct amounts of BTC to each user.

H. Bitcoin Mining Pools and Payment Methods

Although solo miners can compete in the mining process, the likelihood of a successful return is very low. This is even the case for solo miners with the most powerful computing machines. As a solution to this problem, solo miners collaborate in the mining process by joining computational power into mining pools. Together, they form a large organization with significant computational power that can compete with the other large entities. The members of the mining pool work together to find the correct nonce for a candidate block and report the result as one miner, increasing their chances of winning the competition. In the event of success, the rewards are split among the participating miners based on the contribution provided by each.

The concept of a mining pool can be compared to the lottery. Assuming individuals with the same financial capabilities, if a large group buys tickets together, the individuals within the group have a better chance of winning than a single individual buying tickets alone. If any ticket owned by the group wins the lottery, the participating individuals split the reward proportional to the amount invested by each. In a mining pool, the computational power provided by each solo miner is analogous to the amount invested by each ticket buyer.

A mining pool is managed by a pool operator who handles the entire pool server and receives a percentage of the rewards as compensation. The role of the operator is to coordinate the mining performed by all participating miners. The operator keeps a continuously updated copy of the entire blockchain to ease the job of the participating miners. Using the updated blockchain, the operator verifies any transaction that appears in the network and places it in a candidate block for mining. By that, miners only need to worry about finding the correct nonce of the candidate block. If the mining pool wins the competition, the operator divides the rewards among the participating miners.

Reward splitting can be performed in multiple forms and varies from one mining pool to the other. As described in [18], these methods can be categorized into simple reward, score-based reward, or risk-free pay-per share (PPS) reward.

Simple reward systems consist of either proportional systems or PPS systems. In the proportional systems, a reward B is split among the participating miners at the end of each round, where a round is a consecutive time between two successful blocks generated by the pool. The operator keeps a percentage of the reward fB and divides the remaining $(1-f)B$ among the miners based on the shares they submit. Shares are defined as the number of hashes performed by each miner in an attempt to find the correct PoW. A miner that submits n shares from a total of N shares submitted by all the miners in the pool receives a reward of $(n/N)(1-f)B$ BTC on average. Conversely, the PPS system is a deterministic one where the miner knows how much reward can be turned in advance. The operator immediately pays each miner based on the submitted shares regardless of the mining result. In other words, a miner that submits n shares receives $(1-f)pB$ BTC/share, where p represents the probability of one share being the correct PoW. In this system, the operator is taking the risk of mining independently since the miners receive ensured payments whether or not the pool generates a block.

Score-based reward systems come in many forms and strive to prevent miners from pool hopping. Pool hopping is the practice of mining in a pool only during its good times (successfully generating blocks) and leaving it during its bad times. A pool-hopper can maximize his/her rewards at the expense of miners that remain loyal to the pool at all times. The method introduced by Pool [19] is one of the first implemented score-based systems that extends the proportional method. Rather than paying the miner an amount based on the submitted shares after each round, the miner is given a score that is proportional to his/her contribution and increases as more time elapses from the start of the round. The score is used to calculate the reward share given to the miner at the end of the round. However, this method is still susceptible to hopping since the score does not consider factors, such as the mining difficulty or the hashrate of the pool. Also, in this method, mining at the beginning of a round is more profitable since there are fewer shares at that time. As a result, the geometric method was introduced to address these weaknesses. This method introduced a fixed fee, a constant amount taken from the reward of each block, and a variable fee, a score granted at the beginning of each round to the operator. As time passes, the variable fee declines, making mining equally profitable throughout the entire round. Shorter rounds result in larger variable fees and *vice versa*. By that, there is no advantage to mining early in the round.

Another score-based method is pay-per-last-N-shares (PPLNSs) that exists in different forms. In this method, the concept of rewarding miners after each round is replaced with rewarding miners that have been participating in earlier rounds, regardless of the mining result. In other words, the operator pays miners based on their contributions from previous efforts. Later on, more advanced payment systems evolved such as the double geometric method (DGM). This system is a hybrid between the PPLNS and geometric system that combines the advantages of both methods.

Some mining pools employ a risk-free PPS system. One of the first implemented systems is known as the maximum PPS (MPPS). It combines both the PPS and proportional systems, where each participating miner has a balance of each. If the miner submits a share, the PPS balance is incremented and when the pool successfully generates a block, the proportional balance is incremented. At pay time, the miner receives the minimum of both balances. This method protects the pool from taking the risk alone. However, this method is inconsiderate to the miners since they will always make less whether the pool is successful or not. In addition to this, the system suffers from pool hopping. A solution was later proposed to solve this problem in the shared maximum pay-per-share (SMPPS) system. The miners have a PPS balance which continues to accumulate as the miners are participating. If a block is found by the pool and there are sufficient funds, the miners are paid based on their PPS balance. However, if there are no sufficient funds, miners are paid proportional to the available funds and given credit to be paid later for whatever balance that is owed.

Today, a broad range of mining pools exist that give miners a variety of options when joining pools. The question most miners would ask is which mining pool is the best to join. The answer here lies in the preferences of the miners. For example, some miners are not willing to take the risk of not getting paid in the event of being unsuccessful in generating a block and would prefer a PPS mining pool. Others might be willing to take the risk and choose a score-based system for instance, in return for a larger profit.

I. Alternative Cryptocurrencies

In literature, alternative cryptocurrencies are known as *altcoins*, most of which are inspired by Bitcoin. Altcoins strive to offer innovative features and/or enhanced security/privacy countermeasures in an effort to compete with Bitcoin. Their development process is based on the level of innovation and security/privacy countermeasures they present.

The simplest method to develop an altcoin is by forking the open-source code of Bitcoin [20] while adding or modifying any features to it. In software development, a *fork* is a completely independent project that exploits a copy of the original source code. A Bitcoin fork generates an entirely new blockchain and is completely independent of Bitcoin. Namecoin [21] is the first developed Bitcoin fork that adopted all of the characteristics of Bitcoin. It also introduced an additional feature allowing users to store data within its transactions. Various Bitcoin forks have evolved latterly with more features and handled security/privacy issues. Many of these forks implemented privacy protocols to increase the anonymity of cryptocurrencies. In Section VI, we discuss notable privacy protocols that have impacted some of these altcoins.

On exceptional occasions, an altcoin can also be the result of a *hardfork*. A hardfork occurs when modifications are made to the original software of Bitcoin making its new transactions/blocks incompatible with those previously generated prior to the modifications. These modifications can be as simple as altering certain parameters, such as the block size, or as complex as changing major protocols, such as the consensus algorithm. In order to enforce these modifications, the majority of users/miners must upgrade their client nodes to the latest version which accommodates these changes. The users/miners that do not accept the upgrade will view the new transactions/blocks as invalid and will not accept them. As a result, the blockchain will inevitably split into two paths, one storing transactions of the original cryptocurrency and one storing transactions generated due to the modifications made, hence creating a new altcoin. Users in possession of the original cryptocurrency will automatically be granted an equivalent amount of the new altcoin to what they hold.

Bitcoin Cash is a notable example of a Bitcoin hardfork which occurred on August 1, 2017. It was the result of enforcing BIP91 [22] which proposed activating segregated witness (SegWit) [23]. SegWit increases the transaction speed of Bitcoin by splitting the transaction into segments and removing the unlocking signatures which are attached separately at the end. The majority of the miners accepted this proposal resulting in Bitcoin Cash. Users who possessed BTC were immediately granted an equivalent amount of BCC (the currency of Bitcoin Cash) to the BTC they possessed.

While only borrowing the concept of storing transactions in a blockchain, some altcoins have been implemented from scratch with a completely different design and purpose. These altcoins strive to provide services and security/privacy countermeasures beyond the capabilities of Bitcoin or any of its forks. They present substantial differences, such as integrating enhanced consensus algorithms or utilizing private (permissioned) blockchains. In contrast to the public (permissionless) blockchain of Bitcoin, where all participating nodes are allowed to execute the consensus protocol, a private (permissioned) blockchain is limited to only specific nodes. As a result, the cryptocurrency market has witnessed a considerable number of altcoins with substantial innovative features.

III. Bitcoin Security—Double-Spending Attacks

Double spending is an attack that could be performed by malicious users attempting to deceive the system by spending the same BTC more than once. The attacker generates duplicates of the same UTXO and uses it as an input in more than one transaction. Differentiating between the duplicated (fraudulent) copies and the original becomes an issue when used in a decentralized system. There is no trusted entity that verifies the legitimacy of the UTXO used as input in a transaction. The inputs of a transaction may consist of unidentifiable fraudulent BTC that have possibly been spent earlier.

The system defends against such attacks by relying on its users (miners) to validate the legitimacy of the BTC used as an input to transactions. Using the information stored in the blockchain from the previous transactions, the miners validate the inputs of any new transaction to ensure that it does not contain previously spent inputs. Once verified, the transaction is mined into a block that is attached to the blockchain. Any user that refers to the blockchain becomes aware that specific UTXO(s) have been spent earlier, making fraudulent input transactions detectable.

To ensure that attackers cannot manipulate the blockchain in their favor, the mining process is designed to be an expensive and resource-intensive operation. To mine a block of transactions in the blockchain, the miners must provide a valid PoW. An attacker that wishes to double spend BTC must reverse a transaction that has been stored in the blockchain to reuse its inputs in another transaction. Reversing an already stored transaction in the blockchain is an extremely difficult task since it requires a significant share of the total computational power of the system.

In the rest of this section, we will analyze the double-spending attacks. We first discuss conventional methods to perform double spending. Next, we analyze the probability and profitability of the double-spending attack and present a trade-off between the waiting time before accepting a transaction versus the profitability of the attack.

A. Types of Attacks

A double-spending attack comes in many forms. We discuss various techniques that can be performed.

1) Race Attack: A race attack refers to the case where a merchant accepts an unconfirmed transaction (a transaction in a transaction pool waiting to be mined and stored in the blockchain) and immediately provides the payer with a product/service before waiting for confirmation. An attacker with the intention of deceiving the merchant creates two transactions: 1) a transaction that pays the merchant an amount of BTC in return for a product/service and 2) a fraudulent transaction that pays the same amount to the wallet of the attacker. Both transactions use the same inputs (duplicated BTC) and try to spend the same BTC. The attacker concurrently releases both transactions into the Bitcoin network. The miners consider both transactions as being valid until one of them gets stored in the blockchain. The transaction that gets stored in the blockchain is referred to as a confirmed transaction. At that point, the inputs of the stored transaction cannot be used as inputs to other transactions. Therefore, the fraudulent transaction has a chance of being verified first and added to the blockchain making the merchant-paying transaction invalid. The invalid transaction is rejected by the system and dropped from the transaction pools of miners.

To avoid a race attack, merchants must wait for the mining to be completed and the transaction to appear in the blockchain before providing the payer with the product or service. It is recommended that the merchant should wait for at least six subsequent blocks as confirmation before making the trade. In this case, the chances for an attacker to reverse a transaction are negligible, assuming that the attacker can control no more than 10% of the total computational power used in mining.

2) Finney Attack: Finney attack was first suggested in a Bitcoin forum [24]. Similar to the race attack, the attacker performing this attack will only succeed if the merchant accepts an unconfirmed transaction. The attacker creates two transactions similar to those in the race attack and holds on to both of them. The attacker then begins mining the block containing the fraudulent transaction. If the attacker is successful in mining the block, the attacker then uses the other transaction to pay a merchant immediately in exchange for a product/service. Once the merchant makes the trade, the attacker releases the mined block which contains the fraudulent transaction into the network. Given that the block is already mined, it will be added to the blockchain immediately. As a result, the merchant-paying transaction will become invalid. In addition to this, the attacker is rewarded with the mining reward for the mined block carrying the fraudulent transaction. However, the ability to independently mine a block is improbable given the resources necessary to perform the task.

3) Vector76 Attack: In comparison to the race and Finney attacks, the Vector76 attack requires the merchant to wait for a single block to be mined and added to the blockchain as a confirmation. To reverse the transaction, the attacker needs to create a fork in the blockchain. Initially, the attacker creates a merchant-paying transaction and does not broadcast it to the network. Next, the attacker tries to independently and secretly mine this transaction into a block. If successful, the attacker holds onto the block until the honest miners discover another block. The attacker then simultaneously releases the block into the network at the same time as the honest miners release their block which will result in a fork. Before the fork is resolved, the attacker creates a fraudulent transaction that double spends the same BTC used in the merchant-paying transaction. The attacker then relays the fraudulent transaction to the honest miners that do not have the path of the blockchain that carries the merchant-paying transaction. These miners see the fraudulent transaction as valid and begin mining it into a block. As a result, each path of the blockchain stores one of the transactions. If the path that holds the fraudulent transaction grows longer than the other path, the double-spending attempt is successful.

4) 51% Attack: 51% attack is the largest threat to the BTC system. This attack is also referred to as the majority attack in which the attacker (usually a pool of miners) controls more than half of the total computational power of the system. By controlling the majority of the power, the attacker is capable of interfering with the process of mining blocks and reversing any block of transactions. During a 51% attack, the system loses integrity since the other miners no longer have an incentive to compete in the mining process.

To better comprehend this attack, consider the case where the attacker generates a merchant-paying transaction and releases it into the network. The merchant waits for an appropriate number of confirmations before accepting the payment and making the trade. Simultaneously, the attacker secretly begins to mine a block that contains a fraudulent transaction followed by more blocks to extend it. Since the computational power of the attacker is more than the rest of the computational power of all the miners combined, the attacker can mine blocks in less time. Once the merchant accepts the transaction, the attacker releases the secretly mined blocks to create a fork in the blockchain. If the fraudulent fork created by the attacker is longer than the original chain, it becomes dominant and all miners begin to extend on it. By that, the merchant-paying transaction no longer exists in the blockchain.

This attack represents the biggest threat to Bitcoin as it is directly correlated to the resources an attacker can provide. Resources are measured in terms of financial and computational power. Large entities, such as governments or intelligence agencies have the means to control a large share of the total computational power. They are able to destroy or push the system into their favorable status. It is important to note that even with a computational power that is slightly less than 50%, an attacker may still be able to severely manipulate the system. In the next section, we analyze the chances of success of the attackers based on the share of computational power they control.

B. Probability of Success

Despite the continuously increasing popularity of Bitcoin, the number of merchants that have accepted it as a method of payment today is still relatively minimal. Many merchants have concerns about its capabilities in terms of security, while others consider it as a slow method to make payments. Those that accept it should try to take all precautions before accepting a transaction to prevent double-spending attacks.

One of the important precautions is to decide when to accept a transaction before making the trade. Merchants prefer to obtain a certain degree of confidence as assurance that the payer will not be able to reverse the transaction. Those that can afford to wait a long period of time before accepting a transaction (for example, online platforms) require a minimum of six confirmations before accepting a transaction and considering it as being irreversible. However, others that cannot afford this time waiting (such as vending machines), rush into accepting transactions at the risk of losing the payment to a double-spending attack.

Similar to the analysis in [1], we model the race between the honest miners and the attacker to generate blocks as a binomial random walk. The race is denoted as z which represents the number of blocks generated by the honest miners with computational power p minus the number of blocks generated by the attacker with computational power $q = 1 - p$. If a block is generated by the honest miners, we increment z by 1. Conversely, if a block is generated by the attacker, we decrement z by 1. The race between the honest chain and the chain generated by the attacker can be derived as

$$z_{i+1} = \begin{cases} z_i + 1, & \text{with probability } p \\ z_i - 1, & \text{with probability } q \end{cases} \tag{7}$$

where i represents an individual block race. If $q > p$ and the attacker has unlimited resources, the attacker will eventually reach $z < 0$. At that point, the attacker can replace the blocks generated by the honest miners and succeed in performing the attack.

The probability of the attackers to catch up and surpass the blocks generated by the honest miners can be compared to the gambler's ruin problem. Similar to the description in [25], we assume a gambler (attacker) begins with an initial fortune i, $0 < i < N$, and either wins \$1 with probability q or loses \$1 with probability $p = 1 - q$, in each successive gamble. The game represents a random walk which terminates at $i = 0$ (fail) or at $i = N$ (success). The probability of success after i trials is denoted as P_i and can be calculated as

$$P_i = qP_{i+1} + pP_{i-1}. \tag{8}$$

Since $q + p = 1$, we can rewrite (8) as

$$P_{i+1} - P_i = \frac{p}{q}(P_i - P_{i-1}). \tag{9}$$

At $i = 0$, the attacker has a probability of success $P_0 = 0$. By rearranging and generalizing (9), we have

$$\begin{aligned} P_{i+1} &= P_1 \sum_{j=0}^{i} \left(\frac{p}{q}\right)^j \\ &= \begin{cases} P_1 \frac{1-\left(\frac{p}{q}\right)^{i+1}}{1-\left(\frac{p}{q}\right)}, & \text{if } p \neq q \\ P_1(i+1), & \text{if } p = q = 0.5. \end{cases} \end{aligned} \tag{10}$$

Let $i = N - 1$ meaning that $P_N = 1$, we can rewrite (10) as

$$1 = P_N = \begin{cases} P_1 \frac{1-\left(\frac{p}{q}\right)^{N}}{1-\left(\frac{p}{q}\right)}, & \text{if } p \neq q \\ P_1 N, & \text{if } p = q = 0.5. \end{cases} \tag{11}$$

Solve P_1 from (11) and substitute the result into (10) to obtain

$$P_i = \begin{cases} \frac{1-\left(\frac{p}{q}\right)^{i}}{1-\left(\frac{p}{q}\right)^{N}}, & \text{if } p \neq q \\ \frac{i+1}{N}, & \text{if } p = q = 0.5. \end{cases} \tag{12}$$

Following the analysis in [26], we assume that the attacker begins with an initial fortune $i = y$ and can afford to lose up to y dollars before giving up. The gambler wins if $i = N = y + z + 1$ dollars. This assumption modifies the game to account for the probability P_s of the attacker to surpass the blocks generated by the honest miners as

$$P_i = \begin{cases} \frac{1-\left(\frac{p}{q}\right)^{y}}{1-\left(\frac{p}{q}\right)^{y+z+1}}, & \text{if } p \neq q \\ \frac{y+1}{y+z+1}, & \text{if } p = q = 0.5. \end{cases} \tag{13}$$

Consider an attacker that possesses an unlimited amount of resources and is willing to use as much of it as needed to perform the attack, i.e., $y \to \infty$. If $q > p$, then

$$\lim_{y\to\infty} \frac{1 - \left(\frac{p}{q}\right)^{y}}{1 - \left(\frac{p}{q}\right)^{y+z+1}} = 1. \tag{14}$$

For $q < p$, we first divide the numerator and denominator by $(p/q)^y$ then calculate the limit as

$$\lim_{y\to\infty} \frac{\left(\frac{p}{q}\right)^{-y} - 1}{\left(\frac{p}{q}\right)^{-y} - \left(\frac{p}{q}\right)^{z+1}} = \left(\frac{q}{p}\right)^{z+1}. \tag{15}$$

Finally, we can summarize the probability of the attacker to surpass the blocks generated by the honest miners as

$$Q_z = \begin{cases} (q/p)^{z+1}, & \text{if } q < p \text{ or } z \geq 0 \\ 1, & \text{if } q > p \text{ or } z < 0. \end{cases} \tag{16}$$

The merchant has no way of figuring out the number of blocks that the attacker has been able to secretly mine. Therefore, one way to model the overall probability of the attacker to surpass the honest chain is by using the Poisson distribution. The expected number of blocks an attacker can generate is $\lambda = (z+1)(q/p)$. The overall probability P_s of the attacker to surpass the honest chain can be computed by multiplying the Poisson density and the probability of surpassing the honest $z - k$ remaining blocks as discussed in

$$\begin{aligned} P_s &= \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \times Q_{z-k} \\ &= 1 - \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} \begin{cases} 1 - \left(\frac{q}{p}\right)^{z-k+1}, & \text{if } q < p \text{ or } k \leq z \\ 1 - 1 = 0, & \text{if } q > p \text{ or } k > z. \end{cases} \end{aligned} \tag{17}$$

For (17), if $q > p$, we will always have $P_s = 1$, meaning that the attacker will win. When $q < p$, the probability for the attacker to succeed is

$$P_s = 1 - \sum_{k=0}^{z+1} \frac{\lambda^k e^{-\lambda}}{k!} \times \left(1 - \left(\frac{q}{p}\right)^{z-k+1}\right). \tag{18}$$

Another way to model this probability is by using the negative binomial distribution assuming the attacker can premine one block before broadcasting the merchant-paying transaction to the network [27]. The merchant waits for n blocks to be generated by the honest miners with computational power p before accepting the transaction. At that time, the attacker can secretly generate m blocks with computational power $q = 1 - p$, where $m = n - z - 1$. By definition, we can model this as the m number of blocks that the attacker can generate (success) before the n number of blocks the honest miners can generate (failure). Therefore, the probability of a successful double-spending attack for a given value m can be calculated as

$$P(m) = \binom{m+n-1}{m} \times p^n q^m. \tag{19}$$

Overall, the probability for an attacker to successfully surpass the number of blocks generated by the honest miners can be computed as

$$\begin{aligned} P_s &= \sum_{m=0}^{\infty} P(m) \times Q_{n-m-1} \\ &= 1 - \sum_{m=0}^{\infty} \binom{m+n-1}{m} \times p^n q^m \\ &\quad \times \begin{cases} 1 - \left(\frac{q}{p}\right)^{n-m}, & \text{if } q < p \text{ or } k \leq n-m \\ 1 - 1 = 0, & \text{if } q > p \text{ or } k > n-m. \end{cases} \end{aligned} \tag{20}$$

Similar to the previous analysis, (20) confirms that when $q > p$, the attacker will always succeed since $P_s = 1$. When $q < p$, the probability of success can be defined as

$$\begin{aligned} P_s &= 1 - \sum_{m=0}^{n-1} \binom{m+n-1}{k} \times p^n q^m \times \left(1 - \left(\frac{q}{p}\right)^{n-m}\right) \\ &= 1 - \sum_{m=0}^{n-1} \binom{m+n-1}{m} \times \left(p^n q^m - p^m q^n\right). \end{aligned} \tag{21}$$

Fig. 3 shows the results of P_s as n changes based on (21). From this figure, the merchant can obtain the desired level of confidence before accepting a transaction. The obtained level of confidence is definite for any q at $n = 0$, meaning that the attackers have 100% chance of success. As the number of blocks n increases, the chances of a successful double-spending attack decline. Conversely, as q increases, the chances of a successful attack increase. The figure also shows that if $q \geq 0.5$, then we will always get $P_s = 1$. This is known as the majority attack. In fact, even if the values of q are slightly less than 0.5, the chances of a successful double-spending attack could still be high. However, the probability declines exponentially as the value of n increases.

C. Attack Profitability

A successful double-spending attack is only profitable if the revenue is higher than the cost of performing the attack. Suppose an attacker tries to double spend v BTC paid to a merchant in exchange for a product/service. The attacker releases

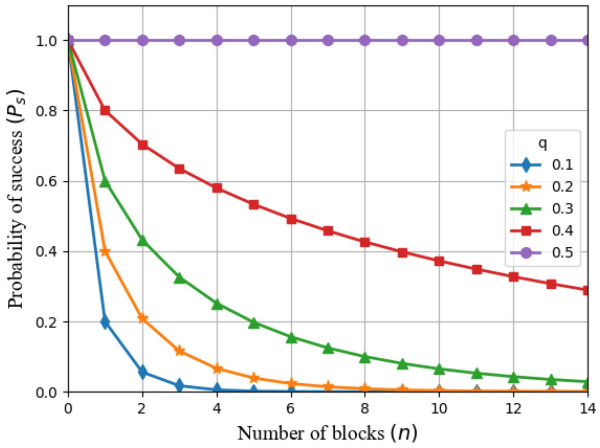

Fig. 3. Probability of successful double-spending attacks versus number of confirmations waited by the merchant.

a transaction into the network that pays v BTC to the wallet possessed by the merchant. Immediately after releasing the transaction, the attacker secretly begins to mine blocks of transactions. One of these blocks contains a fraudulent transaction that pays the same v BTC to the wallet possessed by the attacker. The merchant accepts the transaction after observing that n blocks have been extended to the blockchain. If the attacker is able to secretly mine $m = n + 1$ blocks and replace the n blocks in the blockchain generated by the honest miners, then the attacker is successful in gaining a product/service without paying for it. Assume that the attack returns a value of $2v$, one of which is the actual BTC as a result of reversing the merchant-paying transaction and the other as the product/service. In addition to this, the attacker gains the mining reward for each block mined and the transaction fees included in each transaction. Then, the revenue gained by the attacker can be formulated based on his/her corresponding P_s as follows:

$$\text{Revenue} \approx v + P_s(v + Rm) \text{ BTC} \tag{22}$$

where R is the block reward and the transaction fee per block.

Multiple factors can impact the cost, such as the price and depreciation value of machinery used, the cost of electricity, and the amount of BTC being spent in the transaction. However, formulating the cost with all the possible factors is infeasible. To simplify it, we focus our analysis on the cost factors that could change significantly as the attack is performed. These factors include the v BTC an attacker spends in the merchant-paying transaction, the cost of mining m blocks, and the depreciation cost $d(t)$ of the computing device used in BTC at time t. We derive the cost as follows:

$$\text{Cost} \approx v + me_q(t) + d(t) \text{ BTC} \tag{23}$$

where $e_q(t)$ is the estimated mining electrical cost in BTC per block of a miner with a share q of the total computational power of the system. We assume $e_q(t)$ remains constant during the total time T the attack is performed.

We also assume that the average lifespan of the mining equipment is approximately two years. Using straight-line depreciation, $d(t)$ is a negligible value for an attack over

a short period of time. Therefore, we can reduce the cost equation as follows:

$$\text{Cost} \approx v + me_q(t) \text{ BTC}. \tag{24}$$

The time to mine a single block either by the honest miners or the attackers is approximately 10 min. Therefore, we can rewrite (21), (22), and (24) as

$$P_s \approx 1 - \sum_{m=0}^{\frac{T}{10}-1} \binom{m + \frac{T}{10} - 1}{m} \times \left(p^{\frac{T}{10}} q^m - p^m q^{\frac{T}{10}}\right) \tag{25}$$

$$\text{Revenue} \approx v + P_s\left(v + \frac{RT}{10}\right) \text{ BTC} \tag{26}$$

$$\text{Cost} \approx v + \left(\frac{T}{10}\right) e_q(t) \text{ BTC}. \tag{27}$$

The profit/loss can be formulated from (26) and (27) as

$$\begin{aligned}\text{Profit/Loss} &= \text{Revenue} - \text{Cost} \\ &\approx P_s\left(v + \frac{RT}{10}\right) - \left(\frac{T}{10}\right) e_q(t) \text{ BTC}. \end{aligned} \tag{28}$$

Nowadays, to stand a chance in mining Bitcoin, miners merge their computational power into mining pools as discussed in Section II-H. The mining pools combine the computational power provided by the computing machine of each participating miner. Machines are categorized into one of four groups: 1) application-specific integrated circuits (ASICs); 2) field-programmable gate array (FPGA); 3) graphics processing unit (GPU); and 4) central processing unit (CPU). Each group can provide up to a certain computational power. Comparisons of most of those machines are presented in [28] and [29].

Each computing machine consumes electricity differently based on its specifications. Even machines with similar specifications might vary in cost to operate. As a result, formulating the cost of electricity spent by a miner in the mining process becomes challenging.

ASICs have monopolized the mining process due to their incomparable computational power with those of the CPUs, GPUs, and FPGAs. Miners using any computing machines other than ASICs have a negligible chance of competing. Many mining pools do not permit miners with these machines to join their pools. A miner that joins a mining pool with one of these machines would hardly earn BTC in the event that the pool successfully mines a block. This is because rewards are usually divided among the miners based on their contributions as discussed in Section II-H.

Our goal now is to formulate the estimated electrical cost $e_q(t)$ of a mining pool. First, we estimate the total number of miners $N(t)$ based on the total hashrate $H(t)$ of the system at a certain time t as

$$N(t) \approx H(t)/h(t) \tag{29}$$

where $h(t)$ is the average hashrate of a single mining machine involved in mining at time t.

The cost of electricity is measured in cents/kWh and varies based on the end-use sector and time t. End-use sectors include residential, commercial, industrial, and transportation. We denote the average cost of electricity of all sectors at time t as $e_a(t)$. Using the computing wattage w of the machine, the average running cost $c(t)$ of a machine at time t is

$$c(t) \approx e_a(t) \times w \text{ cents/h}. \tag{30}$$

Using (29) and (30), the total cost $E(t)$ for all miners at time t is

$$E(t) \approx N(t) \times c(t) \text{ cents/h}. \tag{31}$$

We know that it takes approximately 10 min to generate one block, i.e., in $T = 1$ h, miners can generate $m = 6$ blocks. Therefore, the total cost $C(t)$ for all miners to generate one block at $T = 10$ min can be estimated as

$$C(t) \approx \frac{E(t)}{6} \text{ cents/10 min (1 block)}. \tag{32}$$

We estimate the average electricity cost $e_q(t)$ of a mining pool based on its computational power q as

$$e_q(t) \approx C(t) \times q \approx \frac{H(t) \times e_a(t) \times w \times q}{6h(t)} \text{ cents/block}. \tag{33}$$

For our simulations, we assume that the total cost of mining blocks $C(t)$ by all miners and computational power q of the mining pool are fixed during the total mining time T. We also assume a mining environment consists of miners using only ASICs such as Antminer S9 since it is one of the most efficient computing machines on the market today. The specifications of this machine are $h = 14$ TH/s and $w = 1.375$ kWh.

Consider an attacker trying to perform a double-spending attack during the period of February 2020. During that period, 1 BTC was equal to approximately \$9643. The total hashrate power was approximately $H = 110$M TH/s and the average cost of electricity for all sectors in the U.S. was approximately 10.45 cents/kWh, based on the data collected by the U.S. Energy Information Administration [30]. Under these circumstances, in Fig. 4, we present the expected profit/loss of double-spending attacks for various computational powers q. For this analysis, we assume the attackers try to double spend $v = 5$ BTC.

In Fig. 4, a point above $y = 0$ represents a profit while one below it represents a loss. The point of intersection of a curve with $y = 0$ represents the break-even point of an attack. The amount of BTC spent to perform the attack at this time is equal to the revenue returned. By analyzing the figure, we attain the following findings.

1) For any value q at $t = 0$, the attacker turns a profit of exactly v BTC. Recall in Fig. 3, for any value q at $n = 0$ (or $t = 0$), $P_s = 1$. The merchant accepts an unconfirmed transaction giving the attacker a theoretically perfect chance to succeed. In this example, the attacker is trying to double spend $v = 5$ BTC resulting in a profit $= 5$ BTC for all values q at $t = 0$.
2) When the merchant waits for n confirmations before accepting a transaction, the attacker is forced to mine blocks in order to create a fork in the blockchain and succeed in the attack. As discussed in Fig. 3, the probability of success is based on the computational power q

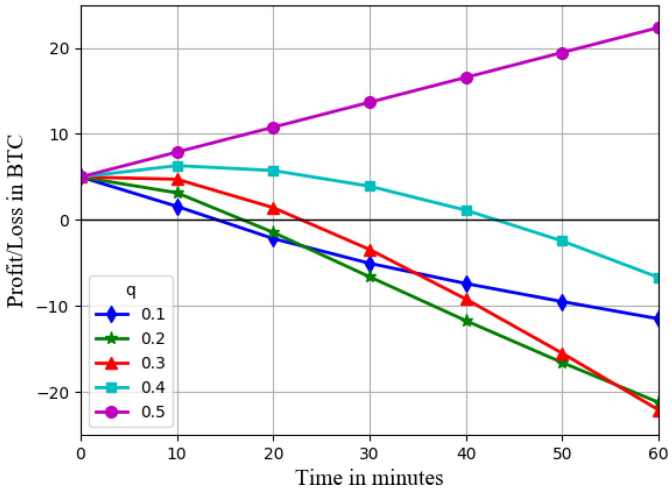

Fig. 4. Profit/loss of attackers with varying computational power q trying to double spend $v = 5$ BTC.

of the attackers. Larger values of q correspond to higher probabilities of success P_s, hence larger profits/losses. We also know that P_s declines as n (or t) increases for all values $q < 0.5$. As a result, the profits eventually turn into a loss as time progresses. An attacker with a smaller value q begins losing at an earlier time during the attack while one with a larger value q can withstand longer periods before losing. However, as reflected by the figure, the losses of attackers with smaller values q are less and continue to increase slower than those with larger values q. This is due to the fact that the cost of electricity $e_q(t)$ for attackers with larger values q are larger than those with smaller values q.

3) In Fig. 4, an attacker with $q = 0.1$ to $q = 0.2$ represents the scenario that begins at the maximum possible profit if only one block is added, then continues to decline till the break-even point. When the number of blocks is increased to two, the attacker will most likely fail to turn a profit. In other words, one block of confirmation for a transaction worth of 5 BTC should give the merchant enough confidence that an attacker with $q = 0.1$ to $q = 0.2$ will not be able to reverse the transaction. If the attacker continues to perform the attack beyond this point, the cost will continue to increase while the revenue declines leading to a loss. As a result, the attacker would most likely surrender at the break-even point to minimize any losses.
4) For $q = 0.3$ to $q = 0.4$, Fig. 4 shows that the profit continues to grow as t increases until it reaches a maximum point due to the accumulation of the mining rewards. Once the chance of success P_s starts to decline with t, the profit also begins to decrease until it reaches the break-even point and later turns into a loss. However, for $q \geq 0.5$, the attacker always succeeds. The profit is represented as a straight line with a positive slope where the slope represents the rate of turning a profit.

As discussed previously, to increase their chances of winning in the mining competition, miners merge their computational power. However, an attacker with a computational power $q < 0.5$ will eventually lose at some point as t increases. On the other hand, an attacker with computational power $q \geq 0.5$ will always succeed with a profit. However, it is important to note that this analysis does not include the *luck* factor. Consider two miners with computational powers q_1 and q_2, respectively, where $q_1 > q_2$. The miner with computational power q_1 has more resources to solve the PoW, therefore, can perform mining faster than the miner with computational power q_2. However, the miner with computational power q_2 could still find the solution to the PoW first due to the randomness of the exhaustive search performed. From a probabilistic standpoint, the chances are low. Moreover, it is worth mentioning that attackers may improve their chances through bribery or selfish mining. In a bribery attack [31], the attacker may purchase mining power from other miners at a premium cost. In a selfish mining attack [32], rational miners are encouraged to join larger mining pools as it would result in larger revenue streams.

IV. Bitcoin Security—Network Attacks

Bitcoin is designed to operate over a P2P network. It is vulnerable to the decentralized network attacks that can escalate other issues. In this section, we will discuss major network attacks that can compromise Bitcoin and present network-related issues. We also suggest possible countermeasures.

A. Denial-of-Service Attacks

Denial-of-Service (DoS) attacks flood the network with bogus traffic in order to disrupt legitimate services and participating components connected to the Bitcoin network. As an example, DoS attacks on a mining pool can result in eliminating the pool from the mining competition, hence giving an advantage to other miners. They could also facilitate double-spending attacks by preventing certain miners from observing the actual transaction flow [33], [34].

Some nodes prefer to privately connect to the Bitcoin network in order to limit the possibilities of becoming victims of DoS attacks. However, this limits the nodes to at most eight outgoing connections. As the number of private nodes increases in the network, the random topology connection weakens. With fewer connections between the nodes, information is flooded at slower rates. There is also no guarantee of the legitimacy of the eight outgoing connections each private node connects to. This means that even a private node can still be vulnerable to a DoS attack if it unluckily connects to malicious nodes.

Bitcoin developers are continuously updating the Bitcoin implementation in an effort to minimize the chances of DoS occurrences. The newer versions analyze the network connections more closely to try to eliminate suspicious nodes from connecting. Developers also strive to limit certain transactions/blocks from being flooded throughout the network. New transactions/blocks are given priority over less important ones such as orphan transactions/blocks. Certain parameters such as block size are also continuously being altered to adjust the

network based on its needs. However, the nature of the P2P network makes Bitcoin vulnerable to these attacks.

B. Sybil Attacks

Peer-to-peer networks are also vulnerable to Sybil attacks [16]. In Sybil attacks, the attacker sets up multiple pseudonymous identities from a single node. In this way, the attacker can acquire an unfair number of shares of the network IP addresses. The honest nodes in the network can easily be deceived into believing that the IP addresses belong to different nodes. With a large number of IP addresses, the attacker can monopolize other connections of nodes and control data propagating to them.

A countermeasure to this attack was proposed in the original Bitcoin white paper [1]. This countermeasure also presented a solution to the majority decision-making problem. It is more convenient to have a one-to-one relationship between a computing machine (node) and a vote instead of having one between an IP address and a vote. An attacker reproducing multiple IP addresses from a single node can no longer make use of them. Every node must engage in a *PoW* procedure to prove its legitimacy as discussed in Section II-F.

Other countermeasures have been taken by the Bitcoin developers to limit Sybil attacks. Each outbound connection is limited to a single IP address per subnet mask 255.255.0.0 (i.e., x.y.0.0/16). In other words, a malicious node can theoretically generate 65 536 IP addresses per network prefix consisting of 16 b where only one can be utilized in a requested outbound connection. Today, owning a machine with different network prefixes that consist of 16 b which can generate numerous IP addresses is impracticable. Malicious users with IP addresses belonging to different network prefixes need to collude in order to pull off such an attack.

Developers can continue to increase the security by limiting outbound connections to larger subnet masks (for example, x.y.z.0/24), however, this would limit the connection possibilities to the outbound connections which contradicts the P2P network. To optimize security, the subnet mask should be modified dynamically based on the available network prefixes of the nodes connected to the network. This optimization is very challenging since there is no fixed pattern to how or when nodes connect to the network. In general, this practice is a weak security countermeasure and can slightly increase security if optimized.

Users should also realize that the majority of node connections are inbound connections (117). If we were to assume that all the eight outbound connections of a node are legitimate, there is no guarantee that the inbound connections are genuine. A private node relies only on its outbound connections to limit its network connections and the data it receives.

C. Eclipse Attacks

The Eclipse attack on Bitcoin was proposed in [35]. The primary purpose of an eclipse attack as defined originally in [36]–[38] is to monopolize all the outbound and inbound connections of a node within a P2P network. As a result, the victim node becomes isolated from the rest of the network and only receives data fed to it by the attacker. By monopolizing the connections of a node, the attacker can control the blockchain view of this node. The eclipse attack targets nodes that are possibly discoverable; nodes with public IP addresses. It strives to populate the tried and new tables of nodes with bogus IP addresses by frequently sending the victim nodes unsolicited addr messages. When the tables of nodes are full, they begin evicting random IP addresses to replace them with the newer ones.

The attack requires the victim node to restart all of its connections. Examples that may cause connection restarts include Internet service provider outages, power failures, or system/software updates. When the node tries reconnecting to its eight permitted outbound connections, it will choose the compromised addresses in either the new or tried tables with a bias toward the newest stored IP addresses. The optimum time to perform the attack is after populating the tables of the victim node with a decent number of controlled IP addresses. The chances of a successful attack are based on the percentage of the controlled IP addresses and the time an attacker spends performing the attack.

To limit an eclipse attack, some countermeasures have been proposed [35]. When replacing IP addresses as newer ones arrive, a deterministic eviction method could be used instead of the random eviction technique. In this way, each IP address is mapped to exactly one slot in the tables rather than multiple slots, requiring the attacker to possess a large number of addresses. Also, allowing random selection of IP addresses rather than choosing the most recent ones when initiating an outbound connection makes the attack less biased to the bogus addresses of the attacker. Other measures include checking an evicted IP address before replacing it with a new one. If the address still connects successfully, there is no reason to evict and replace it with another one. Feeler and anchor connections are also good methods that can disrupt an attacker. Other measures such as increasing the size of the tables, allowing more outgoing connections, or banning unsolicited addr can also greatly limit eclipse attacks.

D. Routing Attacks

The main purpose of a routing attack is to intercept the network transmitted messages and tamper with them. The work presented in [39] proposed a routing attack on Bitcoin via the Internet infrastructure. The border gateway protocol (BGP) [40] is the most widely used protocol when transmitting data between autonomous systems (ASs). An AS manages a set of nodes with similar IP address prefixes and is responsible for routing data between its nodes and other ASs.

The proposed attack intercepts traffic between ASs by performing two independent attacks: 1) *partitioning* attack and 2) *delay* attack. The attack takes advantage of the fact that ASs do not validate the newly announced BGP routes which could result in possible BGP hijacks. A malicious AS can announce forged IP address prefixes to deceive other ASs into believing false routing information. As a result, a successful attacker will be able to intercept all the traffic for nodes with a certain IP address prefix before it reaches its original destination.

The partitioning attack strives to partition the Bitcoin network into two disjoint groups. One group represents the set of isolated nodes while the other group represents the remaining network. The attacker, usually an AS, requires BGP hijacking of other ASs. Once hijacked, the attacker can intercept all inbound and outbound traffic of all the victim ASs. However, the attacker cannot intercept the traffic of stealth connections. Such connections include intra-AS, node connections within the same AS, intrapool, node connections between gateways belonging to the same mining pool, or pool-to-pool, private connections established between pools. Stealth connections can leak data to the isolated group of nodes and result in an attack failure. Therefore, the attacker must detect such nodes and remove them from the lists of nodes to be isolated.

Once the network is divided into two groups, the attacker can perform the delay attack. The main goal of the delay attack is to tamper with data propagating to its destination and cause a stall. The success of the attack relies on the fact that message exchanging (inv, getdata, and tx) is not encrypted. If the attacker intercepts the flow of traffic between ASs, it is possible to tamper with these messages without any node learning about it. For example, when a node within an AS requests data from its peer within another AS, the attacker will intercept the requested message (getdata) and modify the request. As a result, the sending node will send undesired data and cause the receiving node to resend a request message. As long as the attack occurs in a 20-min time frame, the nodes will not lose their connection and will not be aware that their messages are being tampered with.

Apostolaki *et al.* [39] suggested some countermeasures to limit the routing attacks. Simple measures include increasing and diversifying AS connections. Also, monitoring network information such as round-trip time can help identify potential threats. More complex measures can include encrypting messages, using different channels and ports, and simultaneously requesting data from more than one peer. However, implementing such more complex measures could introduce additional cost and delay.

V. Bitcoin Wallet Security

Unlike physical wallets that are used to hold cash and banking cards, Bitcoin wallets behave differently. A bitcoin wallet does not store actual BTC. Instead, it stores the private and public-key pairs that can be utilized to prove the ownership rights to certain BTC stored over the blockchain. As discussed in Section II-B, keys are generated using pseudorandom number generators and elliptic curve cryptography. In this section, we discuss the variations in Bitcoin wallets and outline the security issues in each.

Similar to [41], we first discuss Bitcoin wallet security based on the key generation and infrastructure of the wallet. Three types of wallets have been defined in BTC. We summarize the comparison between all three types in Table I.

The simpler wallets are categorized as nondeterministic wallets, sometimes referred to as *Type-0 wallets*. In these wallets, when a new pair of keys is requested, the wallet generates a random private key as shown in (1). Next, the wallet derives its corresponding public key as described in (1). The generated key pair is completely random and uncorrelated to the previously generated keys. However, these wallets require sophisticated management and could fail to perform well as the number of stored keys grows exceeding the storage capacity of the wallet. A consistent backup of the generated keys is also essential to ensure that the users can still access their BTC in the event of a wallet being unavailable. However, backups are liable to theft and can result in exposing all the keys belonging to a wallet.

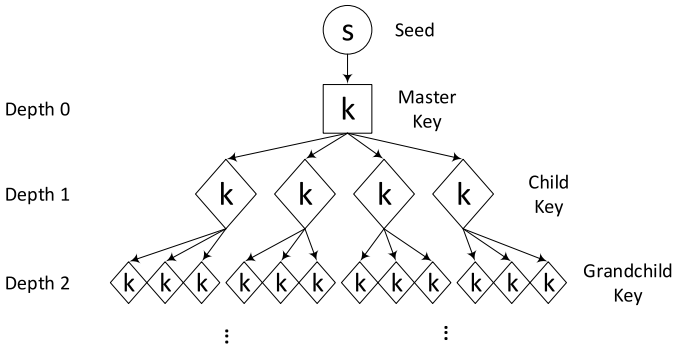

Fig. 5. Structure of a HD wallet.

TABLE I
Wallet Infrastructure Comparison

Infra.	Management	Seed	Backup	Structure
Type 0	Complex	No	All keys	Random
Type 1	Moderate	Yes	Only seed	Sequential
Type 2	Simple	Yes	Only seed	Hierarchical

Deterministic wallets are another type of BTC wallets. They are also referred to as *Type-1 wallets* and can handle the drawbacks of type-0 wallets. In this type of wallet, all the generated keys are based on a common and randomly chosen seed s. Using s, all keys are derived in a deterministic manner. First, a private key with an index i is generated as

$$\mathsf{Pr}_i = \mathsf{SHA256}(s \| i). \tag{34}$$

Using (34), the corresponding public key Pub_i is then generated as discussed in (1). In contrast to nondeterministic wallets, deterministic wallets need only to keep a backup of s to regenerate all of the previously derived keys.

Hierarchical deterministic (HD) wallets, referred to as *Type-2 wallets*, were later introduced based on the BIP0032 standard [42]. In HD wallets, keys are generated in a tree structure as shown in Fig. 5. The key of a node is generated using its corresponding parent node key.

For each node, a key consists of three components: 1) a private key Pr; 2) a public key Pub; and 3) a chain code (CC). The chain code is a third component introduced to prevent the derivation of the key of a child node from only the private and public keys of the parent node. In this way, the *extended key* is an extension of both the private and public keys. The *extended private key* is a combination of the private key and chain code that is used to derive the private key of a child node. Using the derived private key of the child node, it is possible to derive its corresponding public key as explained in (1). On the other

hand, the *extended public key* is a combination of the public key and chain code that is used to derive the public key of a child node. It is important to realize that the public key of a child node can be derived using either the extended private or extended public keys.

The key generation begins at depth 0, which derives the root node (master) key components using a randomly chosen seed (s). In many wallets, s is in the form of a mnemonic word sequence as described in the BIP0039 standard [43]. A mnemonic word sequence is a sequence of English words that represents a random number used to derive s. Using s, the master private key Pr^M and chain code CC^M are derived as

$$\mathsf{Pr}^M = \mathsf{Left256}\big(\mathsf{HMAC\text{-}SHA512}(s)\big) \tag{35}$$

$$\mathsf{CC}^M = \mathsf{Right256}\big(\mathsf{HMAC\text{-}SHA512}(s)\big) \tag{36}$$

where HMAC-SHA512 is a one-way hash-based message authentication code that outputs a 512-b digest and functions Left256 and Right256 extract the left and right 256 b of the digest, respectively. Using the result in (35), the master public key Pub^M is generated as described in (1).

The next step is to generate keys for the children nodes at depth 1 in the tree. Keys can be generated differently depending on the security of the environment in which the wallet is being used. For example, when used in a secure environment, the wallet uses the extended private key to generate all the components of a child node key. This includes the private key that would allow the user to spend BTC from the wallet. Using the private key Pr^p of the parent, we can generate the corresponding public key Pub^p and derive both the private key Pr^c and chain code CC^c of the child using a child key derivation (CKD) function as

$$\mathsf{Pr}^c = \mathsf{Left256}\big(\mathsf{HMAC\text{-}SHA512}\big(\mathsf{Pub}^p \| \mathsf{CC}^p \| i\big)\big) + \mathsf{Pr}^p \tag{37}$$

$$\mathsf{CC}^c = \mathsf{Right256}\big(\mathsf{HMAC\text{-}SHA512}\big(\mathsf{Pub}^p \| \mathsf{CC}^p \| i\big)\big) \tag{38}$$

where $\mathsf{Pub}^p = \mathsf{Pub}^M$, $\mathsf{Pr}^p = \mathsf{Pr}^M$, and $\mathsf{CC}^p = \mathsf{CC}^M$ are the public key, private key, and chain code of the parent node, respectively, and i is the index of the child node. Using the result of (37), we can derive the public key Pub^c of the child node as explained in (1).

On the other hand, when used in an insecure environment, the wallet uses the extended public key to derive only the public key and chain code of the child node instead of the private key. This protects the private key from being exposed to potential attackers. It also allows payments to be made to the wallet while preventing them from being spent. The public key Pub^c and chain code CC^c of the child node are derived using the CKD function as

$$\mathsf{Pub}^c = \mathsf{Left256}\big(\mathsf{HMAC\text{-}SHA512}\big(\mathsf{Pub}^p \| \mathsf{CC}^p \| i\big)\big) + \mathsf{Pub}^p \tag{39}$$

$$\mathsf{CC}^c = \mathsf{Right256}\big(\mathsf{HMAC\text{-}SHA512}\big(\mathsf{Pub}^p \| \mathsf{CC}^p \| i\big)\big). \tag{40}$$

Although using the extended public key is more secure as it does not expose the private key, it may still put the wallet at risk. The extended public key exposes the chain code, which is an essential component in key derivation. Using an exposed chain code and public key, an attacker can perform a brute-force attack on all chain codes derived from it as shown in (40). In other cases, if the private key of a node is compromised in any way, the attacker can use it along with its corresponding exposed chain code to derive the extended private keys of all the descending children nodes as shown in (37) and (38). We also consider the worst-case scenario where an attacker is capable of reversing a derived Pr^c as shown in (37). If successful, using the corresponding parent extended public key, an attacker can derive Pr^p.

To counter these issues, HD wallets also implement an enhanced derivation function known as the *hardened* CKD. This derivation strives to secure the exposed chain code within an extended public key. It prevents the public key of a child node from being derived from the extended public key. Therefore, the extended private key of the parent node is only useful to derive a hardened private key Pr_h^c and chain code CC_h^c of the child node as

$$\mathsf{Pr}_h^c = \mathsf{Left256}\big(\mathsf{HMAC\text{-}SHA512}\big(\mathsf{Pr}^p \| \mathsf{CC}^p \| i\big)\big) + \mathsf{Pr}^p \tag{41}$$

$$\mathsf{CC}_h^c = \mathsf{Right256}\big(\mathsf{HMAC\text{-}SHA512}\big(\mathsf{Pr}^p \| \mathsf{CC}^p \| i\big)\big). \tag{42}$$

Using the result in (41), the corresponding hardened public key Pub_h^c of the child node can be derived as explained in (1). In practice, it is suggested to derive the children keys of the master node using the hardened CKD to keep the master key as secure as possible.

Bitcoin wallets can also take other measures to increase the security of storing keys. Practices, such as P2SH [44] and Multi-Sig transactions increase the security of the BTC stored in the wallet, as discussed in Section II-C. Such techniques are referred to as threshold techniques as they require M-of-N private keys to enable BTC spending. Other wallets enhance the security by encrypting the stored private keys along with a passphrase chosen by the owner of the wallet as defined in the BIP0038 standard [45]. That is

$$\mathsf{Encrypt}(\mathsf{Pr}) = \mathsf{AES}_k(\mathsf{Pr} \| \mathsf{PassPhrase}) \tag{43}$$

where AES is the advanced encryption standard [46] and k is the encryption key. If a user wishes to spend BTC, the user must first decrypt the corresponding encrypted private key using k and the passphrase previously used in encryption. Although encryption provides higher levels of security, the user must keep the passphrase and encryption keys stored securely.

The wallets that exist today come in different forms and account for different security measures. Based on the different installation environments, wallets can be categorized into three types: 1) online (Web) wallets; 2) desktop (software) wallets; and 3) mobile wallets. As in [51], we can further categorize each type of these wallets into: full-service wallets, signing-only wallets, and distributing wallets, based on the functions that they can perform. Table II summarizes these different functions.

A full-service wallet is one that can perform all functions required to spend and receive BTC. These functions include generating private keys needed to spend BTC, signing transactions with the private keys, deriving public keys needed to receive payments of BTC, broadcasting the derived public

TABLE II
WALLET FUNCTION COMPARISON

Type	Function	Examples
Full Service (online)	Generate private key, derive public key, distribute public key, monitor output TX, create/sign unsigned TX, broadcast TX	coinbase.com [47], blockchain.info [48]
Signing-only (offline)	Generate parent private key, derive parent public key, sign TX	Ledger Nano [49], TREZOR [50]
Distributed (offline)	Derive CKD, Distribute public key	customized pre-populated database

keys to the network, and monitoring the BTC spending and receiving of a wallet. Full-service wallets must be able to connect to the Bitcoin network. Examples of online full-service wallets include the wallets provided by coinbase.com [47] and blockchain.info [48]. Armory, Electrum, and Bitcoin core are the most popular desktop full-service wallets today. For mobile wallets, an example that runs on both Android and iOS includes the Airbitz wallet.

The second type of wallets is the signing-only wallets. The main purpose behind these wallets is to enhance the security of the wallet by generating private keys in secure offline environments. Working in conjunction with a networked wallet, the signing-only wallet can interact with the Bitcoin network and can deterministically generate pairs of private and public keys as needed to transfer the public key to the networked wallet. The role of the networked wallet is to distribute the public key to allow payments to be made to the wallet. In case of an HD wallet, the network can also generate child node keys as desired. Once the networked wallet detects a transaction addressed to one of the public keys that it has distributed, it creates an unsigned transaction based on the UTXO and transfers it to the signing-only wallet. The signing-only wallet then uses its private key that could be derived from an extended private key in the case of an HD wallet to sign the transaction and returns it back to the networked wallet. Finally, the networked wallet distributes the signed transaction in the Bitcoin network to claim the BTC.

Signing wallets can either be offline wallets or hardware wallets. Offline wallets are designed to reduce network vulnerabilities. Their tasks include private key derivation and transaction signing. The signed transactions are transferred via removable media to the online wallets. Offline wallets provide higher levels of security than the full-service wallets, however, they require a continuously isolated device. On the other hand, hardware wallets are less of a hassle than offline wallets. They are connected directly to the networked device that eliminates the dependency of removable media when communicating between the signing-only wallet and the networked model. However, the hardware wallet is also inconvenient in situations where the owner makes frequent payments since the owner must constantly carry the hardware wallet to be able to make a payment anytime. As a result, many people use hardware wallets for long-term storage rather than day-to-day transactions. Utilizing this type of wallet, one can store large amounts of BTC in the most secure environments. Popular examples of hardware wallets today include the Ledger [49] and Nano and TREZOR [50].

The final type of wallets is the distributing-only wallets. These wallets also strive to reduce the security issues caused by the full-service wallets. They are in the form of networked wallets for public key distribution in a prepopulated manner, where the public keys are derived and distributed as needed by the network. Other distributing-only wallets are capable of generating the public keys as the case in HD wallets.

Exchange platforms store large portions of cryptocoins in online wallets to provide their users the advantage of reduced transaction time due to the immediate availability of their private keys. This is analogous to storing cash in a centralized entity such as a bank. It is important to point out that storing cryptocoins in an online wallet provided by an exchange platform is the least secure method since it means storing the corresponding private keys that can spend those cryptocoins. The users must completely trust the exchange platform to safely store the private keys and not act maliciously. Even worse, assuming we can trust an exchange platform, cryptocoin owners are still at risk of losing their cryptocoins in the event the exchange platform online wallets are hacked and the private keys are leaked. A hacker that gets a hold of the private keys can immediately use them to send the cryptocoins to his/her personal address. Once the transaction is processed and stored over the blockchain, it becomes immutable to being deleted/modified and most likely will not be reversed unless the blockchain is hard forked.

Throughout the history of cryptocurrencies, multiple attacks have occurred to exchanges that resulted in massive losses and severe price panics to certain cryptocurrencies. In 2011, one of the most notable Japanese-based exchanges, Mt. Gox, online wallets were hacked, leaking all the private keys it stored in the wallet.dat file. Mt. Gox was able to recover from that heist, however, later in 2014, it filed for bankruptcy and was shut down since it was responsible for around 70% of Bitcoin trading volume and lost approximately 850 000 BTC that was valued at more than $450 million. The hackers were able to even steal BTC stored in the exchange's hardware wallets. There is no legitimate evidence of how the attack occurred. In March 2014, Mt. Gox reported on its Website that it had found 200 000 BTC from the total stolen in old-format digital wallets. The other 650 000 were believed to be laundered on another exchange platform known as BTC-e.

The problem is that such heists could possibly occur again. Exchange platforms remain to be extremely attractive hacking points for hackers since they hold so many funds in the least secure manner. Users are recommended to keep limited amounts stored in exchanges while storing the majority of their funds in hardware wallets.

Another issue is whether or not it is possible to track the movement of stolen cryptocoins, hence, catch the hacker. Based on our analysis, it is theoretically possible. However, there have been scenarios where hackers were able to launder large portions of stolen cryptocoins such as the example discussed previously. Another famous example occurred in January 2018 when about $534 million worth of a cryptocoin known as XEM were stolen from a Japanese-based exchange known as Coincheck.

In conclusion, we stress on the fact that there remains to be a tradeoff between the security of a wallet and the ease of use. The most frequently used wallets today are full-service wallets. They are free, user friendly, and can perform all functions needed by a BTC owner. However, these wallets could be vulnerable to theft since they are connected to the network.

VI. Bitcoin Network Privacy

Bitcoin suffers inherent privacy issues in that attackers could link certain identities to their pseudonyms (such as Bitcoin addresses) and identify their history of transactions. This is known as the *linking problem*. Many users publish their real identities and Bitcoin addresses online so that others can make payments to them. This practice is common among blogs and Websites that request BTC as donations or those selling a product or service. These actions could jeopardize their anonymity. Another common example is when users trade BTC for other altcoins over exchange platforms. Most exchange platforms require users to validate their identities by uploading a copy of official identification, which exposes the users to the exchange applications. Such examples do not require an attacker to learn the full transaction history of those users. Simply by tracing the Bitcoin addresses over the blockchain, the transactions could be revealed. In fact, even cautious users who do not publicly use their identities may be at risk as well.

Bitcoin utilizes Bitcoin addresses as its defense mechanisms to preserve the privacy of users. When generated for the users, bitcoin addresses do not leak any information about the identities of the users. However, attackers strive to search for links between bitcoin addresses and user identities using auxiliary information available over the network. If a link is found, it is possible to discover all the other Bitcoin addresses belonging to that user and revealing the complete history of BTC transactions of the user. Today, powerful analysis tools and search engines can be utilized to discover the Bitcoin address and determine this information. Even the strongly encouraged practice of using a new Bitcoin address for every new transaction cannot completely prevent this information from being revealed once a Bitcoin address is linked to an identity of the user.

The auxiliary information can be obtained by multiple methods. Different techniques exist today that can speculate links between Bitcoin addresses and user identities. The study in [52] shows that using information about how nodes are connected within a network can help identify users. In [53], it was shown that patterns of co-occurrences may reveal useful information and lead to any ties. The study in [54] showed that just by monitoring the communication channel, users are likely to lose their anonymity. In [55], an analysis is presented that shows how compromised network nodes can leak significant user information and link them to certain transactions.

Users can run their nodes over Tor [56] in an effort to hide their information from the rest of the network. Tor is a software that provides an additional layer of anonymity. It utilizes multilayer encryption and random relaying nodes to transfer data between a sender and receiver. The sender begins by sending the multilayer encrypted message to a random node that decrypts a single layer and transmits it to the next relaying node. This process continues until the message is completely decrypted and arrives at the receiver [57]. However, multiple studies, such as [58] and [59], have shown that even a low-resource attacker could be capable of gaining information flowing between users running their Bitcoin nodes over Tor. This information can include the data sent between nodes or even the location of the nodes within the network topology.

Other efforts have also been employed in an effort to improve the anonymity of Bitcoin. We classify these efforts into two main classes: 1) mixing services and 2) joint transactions.

A. Bitcoin Mixing Services

BTC mixing is an approach that mixes identifiable BTC in an effort to make them unrecognizable by public observers. The first generation mixing was centralized and performed by *tumblers*. Tumblers are third party mixers that receive BTC from different users, randomly mix them up, and then return to the users their updated BTC amounts. An attacker would no longer be able to trace the BTC of a certain user since the user no longer possesses the same BTC that he/she previously owned. However, a tumbler being a centralized entity presents many threats to the users. It must be fully trusted not to steal the BTC it mixes or even leaks any information about the mixing process. Even when completely trusted, being centralized makes it prone to being compromised. In addition, tumblers charge users mixing fees in return for their services.

In an effort to mitigate these risks, a new generation of decentralized peer-to-peer tumblers was introduced. Instead of sending BTC to a tumbler that performs mixing, the users themselves are involved in the process. For example, CoinSwap [60] is a protocol that allows two parties to exchange Bitcoins with the help of an intermediary who facilitates the Bitcoin swapping. Each party establishes a payment channel with the intermediary using hash time-locked contracts to prevent the intermediary from stealing the Bitcoins. However, given that the intermediary knows both participating parties, it can still compromise privacy. To solve this issue, TumbleBit [61] was later proposed allowing multiple participants to set up payment channels using the same intermediary. Instead of swapping, TumbleBit allows users to make anonymous payments utilizing cryptopuzzles and blind signatures. The paying user would generate a puzzle, blind it, and share the result with the intermediary along with the amount of 1 Bitcoin. The intermediary would then solve the blinded puzzle and return the blinded solution to the paying user. The paying user next unblinds the solution and sends it along with the original puzzle to the receiving user. The receiving user can then share both values with the intermediary and obtain the 1 Bitcoin. As more participants interact with the same intermediary, it becomes infeasible for the intermediary to link the participants together.

B. Bitcoin Joint Transactions

A joint transaction allows different users to combine the inputs and outputs of their transactions into a single transaction to be processed as a whole. All participating users must provide

their own signatures to the transaction to unlock their input portion. Once all participating users correctly sign their inputs, the transaction can be processed as a regular transaction and added to the blockchain. An attacker can no longer trace the BTC movement of a user since there is no direct relationship between the inputs and the outputs of a transaction. The level of privacy provided by a joint transaction increases with the number of participating users. This also results in a lower transaction fee that is paid by each user as it is divided among more users. In 2013, Maxwell introduced this concept as CoinJoin [62] which is widely used in practice today. CoinJoin eventually began to evolve and existed in multiple flavors. Notable examples that introduced new concepts are described as follows.

SharedCoin: Provided by Blockchain.info, SharedCoin is one of the initial implementations of the CoinJoin protocol that ran over a centralized server. The centralized server was the meeting room for the participating users to meet and combine their transactions together. Since users meet in one place, the server is capable of keeping logs of the transactions processed over it. This requires users to completely trust the server not to misuse these logs and put their information at risk if compromised. Shortly, Kristov Atlas created CoinJoin Sudoku, a software that is capable of analyzing the mixing process performed by SharedCoin. The software aims at discovering the relationships between transactions and their owners. It clustered matching inputs and outputs of transactions trying to identify a common owner. However, this implementation is completely suspended due to its various privacy limitations.

Dark Wallet: In 2013, Willson and Taaki introduced *Dark Wallet* [63]. It provides anonymity using stealth addresses and the CoinJoin protocol. A stealth address is a public seed address combined with some metadata used to derive an actual address for a payee to receive transactions. The metadata are shared only between the payer and the payee, and cannot be accessed by the public observers. To generate an actual address, the payee generates a private key and its corresponding public key. Next, the payer uses the public key of the payee and some metadata to generate a transaction with a new address. Once the payee learns the metadata, it can claim the amount attached to the transaction by deriving the appropriate key from the stealth address. Others trying to trace the transaction that was received with a stealth address would not be able to trace it. However, Dark Wallet cannot provide complete anonymity against linking users to certain BTC transactions since the payer can trace it.

CoinShuffle: CoinShuffle was introduced in 2014 [64]. It is a combination of the CoinJoin protocol and the accountable anonymous group communication protocol Dissent [65]. Its main purpose is to eliminate the involvement of third parties while achieving anonymity and protection against DoS attacks. The protocol consists of three main phases: 1) announcement; 2) shuffling; and 3) transaction verification. In the announcement phase, the participants generate a new pair of private and public keys then broadcast their corresponding public key to the other participants. In the shuffling phase, each participant generates a new Bitcoin address to be used as their output address in the mixing transaction. Following that, the participants obliviously shuffle these generated Bitcoin addresses. In the transaction verification phase, every participant checks whether their Bitcoin address is contained in the output list. If present, each participant creates a mixing transaction that spends the inputs to the shuffled list of outputs, signs the transaction, and broadcasts the signature. Once each participant receives the signatures of the others, every participant can generate a fully signed version of the mixing transaction. Dishonest behavior can be detected by the presence of one honest participant who would not broadcast his/her signature and report the dishonesty to all other participants.

However, Coinshuffle suffers anonymity vulnerability if not used cautiously since it allows users to assign change back to themselves in the mixing transaction. Once the change is assigned to the Bitcoin address of the user, anonymity could easily be lost. The best solution to this problem is to use amounts that do not require any change. However, the user does not necessarily get to choose what amount to use since the user must use UTXO(s) from previous transactions. In addition to this, Coinshuffle reveals the identities of the participants among each other during the process.

JoinMarket: JoinMarket [66] is a decentralized CoinJoin implementation. It aimed at improving the privacy of all the previous implementations. JoinMarket introduced two types of participating users, market makers, and market takers. Market makers are users who are willing to mix their BTC at any given time in return for a fee. On the other hand, market takers are users that demand immediate mixing service and are willing to pay a fee as compensation to the market makers. Market makers and takers negotiate the service over an Internet relay chat (IRC) channel. Once terms are discussed, a mixing contract is generated which enables each participating user to operate from their own personal machine. The fact that the system is decentralized protects users from the need to trust a centralized entity. Furthermore, the fee paid by the takers to the makers incentivizes them to continue to join.

VII. Security and Privacy of Altcoins

The continuous emergence of altcoins presents enhanced features to the cryptocurrency enthusiasts. Some of these altcoins have proven to provide enhanced security and privacy over Bitcoin. However, Bitcoin continues to remain at the top of the list of cryptocurrencies with the largest market cap. This contradiction raises questions around its dominance.

In this section, we unfold the major security and privacy advantages of altcoins. We first investigate distinct consensus algorithms implemented by different altcoins in an effort to keep their network secure. We strive to elucidate the security advantages of these algorithms over the PoW implemented by Bitcoin. Next, we discuss major altcoin privacy protocols and privacy improvement over Bitcoin.

A. Altcoin Security

The PoW implemented in Bitcoin utilizes SHA256: a *CPU-bound* function. The time needed to run SHA256 is determined by the speed of the machine. Powerful machines such as ASICs can run SHA256 millions of times faster than various other CPUs, GPUs, and FPGAs. This created an unfair

mining competition since not all miners use the same computing machine. In fact, it eliminates miners using CPUs, GPUs, and FPGAs since their chances of success are negligible when compared to those using ASICs.

This PoW has also been greatly criticized for being an energy-wasting technique. Mining is performed using powerful computing machines that require substantial energy to run. Most of the energy used by all these miners end up being wasted since the output of only one miner is used to extend the blockchain. As a result, the cost of running this PoW to achieve consensus is extremely costly. In addition, it is expected that Bitcoin will suffer a mining tragedy of the commons [67]. The mining reward will converge to 0 since it continues to halve approximately every four years (precisely every 210 000 blocks). Eventually, the miners will no longer have an incentive in taking part in the consensus procedure someday. This will force the transacting users to increase their transaction fees as an alternative incentive to the miners. As a result, both the users and miners will be driven away from the system.

In an effort to mitigate these issues, some altcoins replaced SHA256 with *memory-bound* hash functions in their PoW. In comparison to the CPU-bound function used by Bitcoin, the time needed to run memory-bound functions is determined by the amount of memory available to hold the processed data. Developing ASICs for memory-bound functions is no longer advantageous since they can only optimize CPU-bound functions. Notable examples of such functions include Scrypt [68], CryptoNight [69], Equihash [70], Ethash [71], and Dagger [72]. However, the proposed memory-bound PoW has not been proven to be completely successful. As a result, more effort has been exerted to implement alternative *chain-based* PoW algorithms. For example, combinations of hashing algorithms, such as X11 [73] and X12-X17 [74] have been used in other PoWs where 11–17 different hashing algorithms would be used together and the result of each subalgorithm is fed as input to the next subalgorithm. Other notable chain-based PoW examples also include Lyra2RE [75] and Quark [76] that use five and six different hashing functions, respectively. Unfortunately, the limitations still remain. As a result, consensus algorithms have evolved to pseudorandomly select a single validator to generate the next block of the blockchain. Some widely implemented protocols are described as follows.

Proof-of-Stake (PoS): PoS is an alternative consensus algorithm that was initially suggested in [77]. In contrast to PoW, PoS is dependent on economic stakes of users (i.e., holdings in cryptocurrency) rather than their computational resources. The algorithm deterministically selects a user with significant holdings to validate the next block. In return, the selected validator is rewarded a certain value of the cryptocurrency similar to the mining reward in PoW and all the transaction fees included in the block. Conceptually, a user holding $x\%$ of the total available cryptocurrency will be chosen $x\%$ of the time as the validator in generating the next block. Once the block is generated, the validator relays it to the other validators to confirm it and extend the blockchain.

PoS has multiple benefits in comparison to PoW. Users are no longer required to consume a substantial amount of electricity since they no longer engage in a mining process. In fact, they are motivated to take part in the validation process as it requires nothing more than presenting their wealth in return for a reward if chosen to be the validator. In contrast to PoW, PoS significantly speeds up the consensus process. From a security perspective, PoS tackles the 51% attack by making it more expensive than performing it in a PoW environment. An attacker would need to possess 51% of the total cryptocurrency available to perform the majority attack. Assuming a single user possesses 51% of the total cryptocurrency and performs the attack, the value of the cryptocurrency will drop and the attacker would suffer most, being the majority stakeholder. In comparison to PoW, the majority attack requires 51% of the total mining power that is theoretically achievable through mining pools. This incident previously occurred in the mining environment of Bitcoin as a mining pool (Ghash.IO) exceeded the 51% threshold.

Although PoS could handle some issues caused by PoW, it also introduced some major challenges. The largest stakeholders will be able to monopolize the consensus procedure as they will always be selected and earn the reward. This will create a centralized consensus environment. In addition to this, an attacker with a 51% stake can also completely destroy the cryptocurrency, assuming the intentions of the attacker are to eradicate the system at any price. PoS also suffers a major flaw known as Nothing at Stake (NoS). This issue can occur if coincidentally two stakeholders are chosen to validate the next block. This may result in two valid blocks that can extend the blockchain. As a result, a fork may occur to the blockchain as the miners accept both blocks. To resolve the fork, the validators vote on both branches. Voting is done at no cost which may be an incentive for a malicious validator to vote for a specific path of the blockchain and facilitate a double-spending attack.

These issues resulted in PoS to start appearing in multiple flavors. Its first implementation appeared in a Bitcoin fork, namely, Peercoin [78], which incorporates a *chain-based* PoS, a hybrid of an energy-efficient PoS and the original PoW that runs SHA256. PoW was used initially as a method of coin generation and distribution to get the system running. As time progressed, PoW was slowly replaced by PoS to validate transactions, mint new coins, and maintain consensus. The validators are chosen based on the number of coins in their possession and their corresponding age (i.e., a timestamp indicating how old the coins are). Once they are granted a reward in return for their service, the age of their coins goes back to 0 to give other validators a chance to generate the next block. By that, no single validator can monopolize the validation process.

Later, modified versions of PoS were implemented into some cryptocurrencies. In [79], the age of the coins was removed as it was argued to be abusive to the system. It can help gain significant network weight and facilitate a double-spending attack. In some cases, it may also discourage honest users from staking persistently as they would hold back until their coins are oldest in age to maximize their chances. In addition, Tendermint [80] and casper the friendly ghost (CTFG) [81] introduced a *BFT-style* PoS algorithm. In such a PoS, a validator is selected randomly to propose the next

block, however, they include a process where all validators engage in a multiround process to vote on a specific block at each round. Moreover, in [82], a delegated-style PoS (DPoS) was proposed where the users vote for validators (referred to as witnesses). Each vote has a different strength based on the stake of the user. However, this requires users to completely trust the validators they vote for. Notable examples that implement DPoS include EOS [83] and Tron [84].

Proof-of-Activity (PoA): PoA is a consensus algorithm that combines PoW and PoS into one protocol [85]. Its purpose is to reward only the online participators, thus motivate more miners to remain online in an effort to secure the network. The protocol is analogous to the lottery where the chances of winning an individual are based on the number of tickets the individual holds.

In PoA, miners first utilize their computational power to compete in generating an empty block header; one that does not reference any transactions. A successful miner then immediately broadcasts the resulting hash to the network. This hash value is used to deterministically derive N pseudorandom stakeholders who are potential miners if found to be online. The derivation of these N stakeholders is performed by hashing a concatenation of the broadcast hash value, the hash of the previous block, and N fixed suffix values. The protocol then invokes a subroutine known as *follow-the-satoshi* once for each derived value. The subroutine finds the block storing a satoshi with the same index as the result. Next, it inspects the block in which the satoshi was minted and traces its movement up until its last owner. If online, this owner participates in the next block generation process that extends the blockchain. Similar to PoS, the more satoshis an individual owns, the more likely that the individual will be selected randomly in this process.

Every stakeholder then checks the validity of the empty block header that was initially broadcast. Using this value, they also check whether they were one of the N selected validators. The first $N-1$ lucky stakeholders sign the hash of the empty block header with the private key that controls the satoshi derived from *follow-the-satoshi* subroutine. Next, they broadcast their signature to the network. The Nth stakeholder then generates a wrapped block that extends the empty block header by including the desired transactions to be verified, the $N-1$ signatures, and his/her own signature for this block. The wrapped block is finally broadcast to the network to extend the blockchain. The transaction fees that the Nth stakeholder collects from the included transactions are shared among the miner and the N participators.

From a security perspective, PoA makes the 51% attack more difficult than PoW and PoS since a large computational power and a significant stake are both required in PoA.

Proof-of-Burn (PoB): PoB is an algorithm that achieves consensus by *burning* a portion of a cryptocurrency. Burning a portion of cryptocurrency means generating a transaction with this portion destined to an inaccessible address by all users. The concept of burning is analogous to buying expensive computational hardware in PoW.

In general, a miner burns portions of his/her holdings and waits a certain period of time. This time ensures that it is impractical for an attacker to undo the transaction. After waiting, the transaction is permanently stored in the blockchain and becomes visible to all observers. This is proof that the potential miner has invested a portion of his/her holdings and is worthy of being a miner. Honest miners will burn portions of their holdings that are less than or equivalent to what they can return in the mining process if successful. In other words, if miners burn more than what they are expected to return in a successful mining process, they will spend more than what they earned, hence a loss.

The potential miners then create candidate blocks in an effort to extend the blockchain. By referencing their transactions in the blockchain, they can prove that they have burnt some of their holdings earlier, thus become accepted by the community as miners. The winning block that extends the blockchain is chosen by allocating the miner that has burnt the most after a certain period of time.

From a security perspective, this algorithm can achieve the same security as its predecessor algorithms. It requires a miner to perform an expensive task (burning) that is easily verified by all other participators observing the blockchain. Similar to PoS, it saves the miners the hassle of buying hardware to physically perform mining.

B. Altcoin Privacy

Privacy is one of the most important issues of cryptocurrencies. Below, we describe some notable protocols that have been implemented in altcoins.

Zero-Knowledge Proofs: Zero-knowledge proofs have been implemented into some altcoins to provide private transactions. They allow a prover to convince a verifier that numbers in transactions exist without revealing any information about the actual numbers. Notable variants of such proofs have been implemented, such as zk-SNARKs [86], zk-STARKs [87], and BulletProofs [88]. In comparison to zk-SNARKs, both zk-STARKs and BulletProofs do not require a trusted entity during the setup phase and provide a smaller size of proof, making the size of transactions smaller. However, the verifier computational complexity of zk-SNARKs is lower.

PrivateSend: PrivateSend is an altcoin joint transaction protocol [89] that combines identical inputs from various users into one transaction with multiple outputs. A user initially reaches out to a random master node requesting mixing specific denominations of a certain amount of coins. The master node then announces that it is willing to accept other coins of identical quantities and denominations to be mixed into a transaction. Once the master node receives enough requests, the involved users specify their full list of inputs and outputs they wish to be mixed. The inputs specify the coins to be mixed while the outputs specify the output addresses of users where they wish to receive the mixed coins. The master node then puts all inputs and outputs into a joint transaction and sends it to the involved users. The users validate the transaction and sign their inputs and return it to the master node. The master node finally broadcasts the transaction to the network which is treated as any other transaction. However, PrivateSend is not a completely decentralized protocol and can jeopardize the anonymity of the user since it involves a centralized node.

CryptoNote: CryptoNote is a privacy-preserving protocol embedded in some cryptocurrency implementations that strive to hide the connection between a sender and receiver from the rest of the network [69]. The protocol protects the identity of the sender by utilizing ring signatures [90] when signing transactions. The public key of the sender is shuffled with public keys of other senders, giving all keys equal probability of being linked to a transaction. In this way, an attacker has no way of identifying the private key used during transaction signing, hence identifying the sender. It also generates a unique public key for the receiver with each new incoming transaction. Using random data generated by the sender and the public key of the receiver, a one-time unique pair of private and public keys is generated via the Diffie–Hellman key exchange [91]. These keys are used to claim the transaction output by the receiver. However, since CrypoNote does not reveal the information of transactions, verifying transactions becomes a challenge. To handle this issue, a modified version of the original traceable ring signature [92] is utilized.

C. Mimblewimble

Similar to CryptoNote, Mimblewimble [93] is a protocol that hides connections between senders and receivers. It eliminates the inputs and outputs in the UTXO design model replacing them with a single multisignature known as confidential transactions. The multisignature uses the Pedersen commitment scheme and is generated by both sender and receiver. Both parties share a blinding factor used to encrypt all inputs and outputs of their transaction and their private and public keys. Using the Pedersen commitment, the blockchain nodes can deduct the encrypted amounts from both inputs and outputs to ensure that the coins actually exist. As a result, the transaction amounts remain completely obscured from the entire network, providing privacy to both users.

VIII. Future Research Directions

Following Bitcoin, some blockchain systems began incorporating smart contracts to provide services besides simple payments, for example, Ethereum [94]. A smart contract is a self-executing code that is stored and processed over the blockchain. These contracts allow users to negotiate and come to agreements without needing a trusted third party. They may significantly facilitate payments that require certain prerequisites before initiating transactions. Similar to Bitcoin transactions, when smart contracts are deployed over the blockchain and executed by miners, the result is immutable and irreversible. However, in comparison to Bitcoin, they present new security and privacy concerns as they require faultlessness. Once deployed over the blockchain, smart contracts cannot be altered and will continue to exist indefinitely. Inappropriately implemented smart contracts may fall victims to attacks [95] that may jeopardize the security and privacy of users.

Interestingly, since the outset of smart contracts, various blockchain-based applications began to evolve beyond financial services [96]. The main purpose of these applications is to make use of the permanent, transparent, and irreversible features of the blockchain. These applications are emerging in various domains.

From a commercial standpoint, blockchain could be useful in any industry that applies the supply chain in its manufacturing process. For example, the automobile industry could reduce the number of recalls when a part is found to be faulty by storing all automobile parts and their data as they change with time over the blockchain. Tracing the data of a certain part over the blockchain starting from its origin would then allow a manufacturer to come to conclusions and enhance future decision making. Similarly, blockchain could be used to trace sensor data and mitigate its duplication in IoT applications [97].

The healthcare industry is another popular area that could potentially benefit from the blockchain. Patient data could be stored over the blockchain thus help doctors trace back the entire history of a patient in just a few seconds and assist them with making more efficient treatment decisions [98]–[100]. On larger scales, the stored patient data could even be used to potentially enhance research that focuses on predicting medical patterns among patients.

Governments have also shown interest in the blockchain. A good example is clearly illustrated in applications such as electronic voting where a high degree of transparency and immutability is required [101], [102]. Votes could be stored over the blockchain allowing the tallying process to be performed within seconds. Blockchain can also provide voter verifiability that is an imperative feature to trust election results.

IX. Conclusion

Since the inception of Bitcoin, the cryptocurrency market has grown aggressively. It has also raised serious questions, in particular, are cryptocurrencies the future of monetary systems? We strived to address this enigma from a technical perspective by focusing on the security concerns, privacy issues, and their tradeoffs versus the transaction delay of Bitcoin. These concerns are key factors in determining the future of cryptocurrencies. To address these concerns, we first introduced the background of Bitcoin and explicated its major building blocks and protocols. Next, we delved into crucial security concerns. We began by discussing the double-spending attack and analyzing its probability of success. Using this analysis, we further evaluated the profitability of a potential attack. We showed that anyone in possession of less than half of the total computational power will eventually lose at some point while performing the attack. In addition, we explored the major network-related security issues of the underlying peer-to-peer network. Our discussion showed that these network attacks are inevitable since there is no method to restrict malicious nodes from connecting to the network. We further explored storage security by investigating the wallet infrastructures and the different modes of storage. We concluded that there is a tradeoff between storage security and practicality. Beyond the security issues that Bitcoin suffers, we investigated the privacy limitations inherent to the system. We debunked the misconception of Bitcoin anonymity

and reviewed major methods for privacy protection. We also looked to expand the reader's knowledge of emerging altcoins with advanced security and privacy features.

Extensive research is still needed in the blockchain space to fully understand and enhance the security and privacy of such applications. Though many applications aim at further enhancing security and/or privacy, the users remain skeptical due to various application-specific requirements. As research advances in these fields, it is possible that blockchain-based systems may help revolutionize and supersede the centralized systems we rely on today.

REFERENCES

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

[2] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," in *Proc. Conf. Theory Appl. Cryptography*, 1990, pp. 437–455.

[3] D. Bayer, S. Haber, and W. S. Stornetta, "Improving the efficiency and reliability of digital time-stamping," in *Proc. Sequences II*, 1993, pp. 329–334.

[4] S. Haber and W. S. Stornetta, "Secure names for bit-strings," in *Proc. 4th ACM Conf. Comput. Commun. Security*, 1997, pp. 28–35.

[5] H. Massias, X. S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirement," in *Proc. 20th Symp. Inf. Theory Benelux*, 1999.

[6] G. Karame, E. Androulaki, and S. Capkun, "Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin," in *Proc. IACR Cryptol. ePrint Archive*, 2012, p. 248.

[7] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," *Appl. Innov.*, vol. 2, pp. 6–10, 2016.

[8] L. Dashjr. (2016). *BIP Process, Revised*. [Online]. Available: https://github.com/bitcoin/bips/wiki/Comments:BIP-0002

[9] G. Hileman and M. Rauchs, *Global Cryptocurrency Benchmarking Study*, Cambridge Centre Alternative Finance, Cambridge, U.K., 2017.

[10] (2018). *Coin Market Capital*. [Online]. Available: https://coinmarketcap.com/

[11] D. Chaum, "Blind signatures for untraceable payments," in *Proc. Adv. Cryptol.*, 1983, pp. 199–203.

[12] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Proc. Adv. Cryptol.*, 1990, pp. 319–327.

[13] W. Dai. (1998). *B-Money*. [Online]. Available: http://www.weidai.com/bmoney.txt

[14] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, 1987, pp. 369–378.

[15] D. R. L. Brown, *SEC 2: Recommended Elliptic Curve Domain Parameters*, Certicom Res., Mississauga, ON, Canada, 2010.

[16] J. R. Douceur, "The sybil attack," in *Proc. Int. Workshop Peer Peer Syst.*, 2002, pp. 251–260.

[17] J. Poon and T. Dryja. (2016). *The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments*. [Online]. Available: https://www.bitcoinlightning.com/wp-content/uploads/2018/03/lightning-network-paper.pdf

[18] M. Rosenfeld, "Analysis of bitcoin pooled mining reward systems," 2011. [Online]. Available: arXiv:1112.4980.

[19] S. Pool. (2017). *Reward System*. [Online]. Available: https://slushpool.com/help/manual/rewards

[20] T. B. C. developers. (2017). *Bitcoin/Bitcoin*. [Online]. Available: https://github.com/bitcoin/bitcoin

[21] A. Loibl, "Namecoin," *Network Architectures and Services*, vol. 107, 2014. [Online]. Available: https://www.namecoin.org

[22] J. Hilliard. (2017). *Reduced Threshold Segwit Masf*. [Online]. Available: https://github.com/bitcoin/bips/wiki/Comments:BIP-0091

[23] P. Wuille. (2015). *Segregated Witness and Deploying it for Bitcoin*. [Online]. Available: https://prezi.com/lyghixkrguao/segregated-witness-and-deploying-it-for-bitcoin/

[24] H. Finney. (Feb. 2011). *Best Practice for Fast Transaction Acceptance—How High Is the Risk?*. [Online]. Available: https://bitcointalk.org/index.php?topic=3441.msg48384#msg48384

[25] K. Sigman. *Gambler's Ruin Problem*. [Online]. Available: http://www.columbia.edu/∼ks20/FE-Notes/4700-07-Notes-GR.pdf

[26] A. P. Ozisik and B. N. Levine, "An explanation of Nakamoto's analysis of double-spend attacks," 2017. [Online]. Available: arXiv:1701.03977.

[27] M. Rosenfeld, "Analysis of hashrate-based double spending," 2014. [Online]. Available: arXiv:1402.2009.

[28] (2017). *Mining Hardware Comparison*. [Online]. Available: https://en.bitcoin.it/wiki/Mining_hardware_comparison

[29] (2017). *Non-Specialized Hardware Comparison*. [Online]. Available: https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison

[30] *U.S. Energy Information Administration*. https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a

[31] J. Bonneau, "Why buy when you can rent?" in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2016, pp. 19–26.

[32] I. Eyal and E. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," in *Proc. Int. Conf. Financ. Cryptography Data Security*, 2014, pp. 436–454.

[33] B. Johnson, A. Laszka, J. Grossklags, M. Vasek, and T. Moore, "Game-theoretic analysis of DDoS attacks against bitcoin mining pools," in *Proc. Financ. Cryptography Data Security*, 2014, pp. 72–86.

[34] M. Vasek, M. Thornton, and T. Moore, "Empirical analysis of denial-of-service attacks in the bitcoin ecosystem," in *Proc. Financ. Cryptography Data Security*, 2014, pp. 57–71.

[35] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. USENIX Security Symp.*, 2015, pp. 129–144.

[36] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structured peer-to-peer overlay networks," *ACM SIGOPS Oper. Syst. Review*, vol. 36, no. 1, pp. 299–314, 2002.

[37] A. Singh, T.-W. Ngan, P. Druschel, and D. S. Wallach, "Eclipse attacks on overlay networks: Threats and defenses," in *Proc. IEEE INFOCOM*, 2006.

[38] E. Sit and R. Morris, "Security considerations for peer-to-peer distributed hash tables," in *Peer-to-Peer Systems* (LNCS 2429). Heidelberg, Germany: Springer, 2002, pp: 261–269.

[39] M. Apostolaki, A. Zohar, and L. Vanbever, "Hijacking bitcoin: Routing attacks on cryptocurrencies," in *Proc. IEEE Symp. Security Privacy (SP)*, 2017, pp. 375–392.

[40] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)," IETF, RFC 4271, 2005.

[41] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. New York, NY, USA: O'Reilly Media, 2014.

[42] P. Wuille. (2012). *Hierarchical Deterministic Wallets*. [Online]. Available: https://github.com/bitcoin/bips/wiki/Comments:BIP-0032

[43] M. Palatinus, P. Rusnak, A. Voisine, and S. Bowe. (2013). *Mnemonic Code for Generating Deterministic Keys*. [Online]. Available: https://github.com/bitcoin/bips/wiki/Comments:BIP-0039

[44] G. Andresen. (2012). *Pay to Script Hash*. [Online]. Available: https://github.com/bitcoin/bips/wiki/Comments:BIP-0016

[45] M. Caldwell and A. Voisine. (2012). *Passphrase-Protected Private Key*. [Online]. Available: https://github.com/bitcoin/bips/wiki/Comments:BIP-0038

[46] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*. New York, NY, USA: Springer, 2013.

[47] Coinbase. (2018). *Coinbase—Buy/Sell Digital Currency*. [Online]. Available: https://www.coinbase.com

[48] Blockchain. (2018). *Bitcoin Block Explorer—Blockchain*. [Online]. Available: https://blockchain.info

[49] Ledger. (2018). *Ledger Wallet—Hardware Wallets—Smartcard Security for Your Bitcoins*. [Online]. Available: https://www.ledgerwallet.com

[50] SatoshiLabs. (2018). *Trezor Bitcoin Wallet—The Original and Most Secure Hardware Wallet*. [Online]. Available: https://trezor.io

[51] (2017). *Bitcoin Developer Guide*. [Online]. Available: https://bitcoin.org/en/developer-guide#wallets

[52] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *Proc. 30th IEEE Symp. Security Privacy*, 2009, pp. 173–187.

[53] D. J. Crandall, L. Backstrom, D. Cosley, S. Suri, D. Huttenlocher, and J. Kleinberg, "Inferring social ties from geographic coincidences," *Proc. Nat. Acad. Sci. USA*, vol. 107, no. 52, pp. 22436–22441, 2015.

[54] R. Puzis, D. Yagil, Y. Elovici, and D. Braha, "Collaborative attack on Internet users' anonymity," *Internet Res.*, vol. 19, no. 1, pp. 60–77, 2009.

[55] A. Korolova, R. Motwani, S. U. Nabar, and Y. Xu, "Link privacy in social networks," in *Proc. 17th ACM Conf. Inf. Knowl. Manag.*, 2008, pp. 289–298.

[56] R. Dingledine, N. Mathewson, and P. Syverson, *ToR: The Second-Generation Onion Router*, Naval Res. Lab., Washington, DC, USA, 2004.

[57] J. Ren and J. Wu, "Survey on anonymous communications in computer networks," *Comput. Commun.*, vol. 33, pp. 420–431, Mar. 2010.

[58] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin P2P network," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 15–29.

[59] A. Biryukov and I. Pustogarov, "Bitcoin over Tor isn't a good idea," in *Proc. IEEE Symp. Security Privacy (SP)*, 2015, pp. 122–134.

[60] G. Maxwell. (2013). *CoinSwap: Transaction Graph Disjoint Trustless Trading*. [Online]. Available: https://bitcointalk.org/index.php?topic=321228.0

[61] E. Heilman, L. Alshenibr, F. Baldimtsi, A. Scafuro, and S. Goldberg, "TumbleBit: An untrusted bitcoin-compatible anonymous payment hub," in *Proc. Netw. Distrib. Syst. Security Symp.*, 2017, p. 6.

[62] G. Maxwell. (Aug. 2013). *Coinjoin: Bitcoin Privacy for the Real World*. [Online]. Available: https://bitcointalk.org/?topic=279249

[63] C. Willson and A. Taaki. (2017). *Dark Wallet*. [Online]. Available: https://www.darkwallet.is/

[64] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "CoinShuffle: Practical decentralized coin mixing for bitcoin," in *Proc. Eur. Symp. Res. Comput. Security*, 2014, pp. 345–364.

[65] H. Corrigan-Gibbs and B. Ford, "Dissent: Accountable anonymous group messaging," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 340–350.

[66] A. Gibson and C. Belcher. (2017). *Joinmarket*. [Online]. Available: https://github.com/JoinMarket-Org/JoinMarket-Docs/blob/master/High-level-design.md

[67] G. Hardin, "The tragedy of the commons," *J. Nat. Resources Policy Res.*, vol. 1, no. 3, pp. 243–253, 2009.

[68] C. Percival, "Stronger key derivation via sequential memory-hard functions," in *Proc. Techn. BSD Conf. (BSDCon)*, 2009.

[69] N. V. Saberhagen. (2013). *CryptoNote V.2.0*. [Online]. Available: https://cryptonote.org/whitepaper.pdf

[70] A. Biryukov and D. Khovratovich, "Equihash: Asymmetric proof-of-work based on the generalized birthday problem," *Ledger*, vol. 2, pp. 1–30, Mar. 2017.

[71] J. Ray. (2018). *Ethash*. [Online]. Available: https://github.com/ethereum/wiki/wiki/Ethash

[72] V. Buterin. (2013). *Dagger: A Memory-Hard to Compute, Memory-Easy to Verify Scrypt Alternative*. [Online]. Available: http://www.hashcash.org/papers/dagger.html

[73] B. Kiraly. (2017). *X11*. [Online]. Available: https://dashpay.atlassian.net/wiki/spaces/DOC/pages/1146918/X11

[74] PiMP. (2017). *Blog: What Are All These X11, X13, X15 Algorithms Made of?* [Online]. Available: https://getpimp.org

[75] (2019). *Lyra2re*. [Online]. Available: https://en.bitcoinwiki.org/wiki/Lyra2RE

[76] (2018). *Quark Algorithm*. [Online]. Available: https://en.bitcoinwiki.org/wiki/Quark_Algorithm

[77] Q. Mechanic. (2011). *Proof of Stake Instead of Proof of Work*. [Online]. Available: https://bitcointalk.org/index.php?topic=27787.0

[78] S. King and S. Nadal. (Aug. 2012). *PPcoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. [Online]. Available: https://decred.org/research/king2012.pdf

[79] P. Vasin. (2014). *Blackcoin's Proof-of-Stake Protocol V2*. [Online]. Available: https://blackcoin.org/blackcoin-pos-protocol-v2-whitepaper.pdf

[80] J. Kwon, "Tendermint: Consensus without mining," *Draft v. 0.6, Fall*, vol. 1, no. 11, 2014. [Online]. Available: https://cdn.relayto.com/media/files/LPgoWO18TCeMIggJVakt_tendermint.pdf

[81] V. Zamfir. (2017). *Casper the Friendly Ghost: A Correct by Construction Blockchain Consensus Protocol*. [Online]. Available: https://github.com/ethereum/research/blob/master/papers/caspertfg/caspertfg.pdf

[82] D. Larimer, "DPOS Consensus Algorithm-The Missing White Paper; Delegated proof-of-stake (DPoS)," Bitshare, White Paper, 2014. [Online]: [Online]. Available: https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper

[83] (2018). *Eos.IO Technical White Paper v2*. [Online]. Available: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[84] T. Foundation. (2018). *Tron Protocol Version: 3.2*. [Online]. Available: https://tron.network/static/doc/white_paper_v_2_0.pdf

[85] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] Y," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 42, no. 3, pp. 34–37, 2014.

[86] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a Von Neumann architecture," in *Proc. USENIX Security Symp.*, 2014, pp. 781–796.

[87] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. (Mar. 6, 2018). *Scalable, Transparent, and Post-Quantum Secure Computational Integrity*. [Online]. Available: https://eprint.iacr.org/2018/046.pdf

[88] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *Proc. IEEE Symp. Security Privacy (SP)*, 2018,. pp. 315–334.

[89] B. Kiraly. (2017). *PrivateSend*. [Online]. Available: https://dashpay.atlassian.net/wiki/spaces/DOC/pages/1146924/PrivateSend

[90] R. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *Proc. Adv. Cryptol. ASIACRYPT*, 2001, pp. 552–565.

[91] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. IT-22, no. 6, pp. 644–654, Dec. 1976.

[92] E. Fujisaki and K. Suzuki, "Traceable ring signature," in *Public Key Cryptography* (LNCS 4450). Berlin, Germany: Springer, 2007, pp. 181–200.

[93] T. E. Jedusor. (2016). *Mimblewimble*. [Online]. Available: https://scalingbitcoin.org/papers/mimblewimble.txt

[94] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project, Zug, Switzerland, 2014.

[95] K. Zipfel. (2019). *Smart Contract Attack Vectors*. [Online]. Available: https://github.com/KadenZipfel/smart-contract-attack-vectors

[96] S. Aggarwal, R. Chaudhary, G. S. Aujla, N. Kumar, K.-K. R. Choo, and A. Y. Zomaya, "Blockchain for smart communities: Applications, challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 144, pp. 13–48, Oct. 2019.

[97] I. Makhdoom, M. Abolhasan, H. Abbas, and W. Ni, "Blockchain's adoption in IoT: The challenges, and a way forward," *J. Netw. Comput. Appl.*, vol. 125, pp. 251–279, Dec. 2019.

[98] A. Azaria, A. Ekblaw, T. Vieira, and A. Lippman, "MedRec: Using blockchain for medical data access and permission management," in *Proc. IEEE Open Big Data (OBD)*, 2016, pp. 25–30.

[99] E. Zaghloul, T. Li, M. W. Mutka, and J. Ren, "d-MABE: Distributed multilevel attribute-based EMR management and applications," *IEEE Trans. Service Comput.*, early access.

[100] T. McGhin, K.-K. R. Choo, C. Z. Liu, and D. He, "Blockchain in healthcare applications: Research challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 135, pp. 62–75, Jun. 2019.

[101] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in *Proc. Financ. Cryptography Data Security*, 2017, pp. 357–375.

[102] E. Zaghloul, T. Li, and J. Ren, "Anonymous and coercion-resistant distributed electronic voting," in *Proc. IEEE ICNC*, 2020, pp. 389–393.

Ehab Zaghloul received the B.S. and M.Sc. degrees in computer engineering from the Arab Academy for Science and Technology, Alexandria, Egypt, in 2012 and 2015, respectively, and the Ph.D. degree in electrical and computer engineering with Michigan State University, East Lansing, MI, USA, in 2020.

His research interests include applied cryptography, secure and private cloud data sharing, cryptocurrencies, and blockchain.

Tongtong Li (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from Auburn University, Auburn, AL, USA, in 2000.

From 2000 to 2002, she was with Bell Labs, Murray Hill, NJ, USA, and had been working on the design and implementation of 3G and 4G systems. Since 2002, she has been with Michigan State University, East Lansing, MI, USA, where she is currently an Associate Professor. Her research interests fall into the areas of wireless and wired communications, wireless security, information theory, and statistical signal processing with applications in neuroscience.

Dr. Li is a recipient of the National Science Foundation CAREER Award for her research on efficient and reliable wireless communications in 2008. She served as an Associate Editor for IEEE SIGNAL PROCESSING LETTERS from 2007 to 2009, and an Editorial Board Member for the *EURASIP Journal Wireless Communications and Networking* from 2004 to 2011. She served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING from 2012 to 2016.

Matt W. Mutka (Fellow, IEEE) received the B.S. degree in electrical engineering from the University of Missouri-Rolla, Rolla, MO, USA, the M.S. degree in electrical engineering from Stanford University, Stanford, CA, USA, and the Ph.D. degree in computer sciences from the University of Wisconsin–Madison, Madison, WI, USA, in 1988.

He is a Professor on the Faculty of the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA, where he is currently serving as the Program Director with the National Science Foundation, Division of Computer and Networks Systems, Directorate for Computer and Information Science and Engineering. He served as a Chairperson with the MSU Department of Computer Science and Engineering from 2007 to 2017. He has been a Visiting Scholar with the University of Helsinki, Helsinki, Finland, and a member of the Technical Staff with Bell Laboratories, Denver, CO, USA. His current research interests include mobile computing, sensor networking, and wireless networking.

Prof. Mutka was honored with the MSU Distinguished Faculty Award.

Jian Ren (Senior Member, IEEE) received the B.S. and M.S. degrees in mathematics from Shaanxi Normal University, Xi'an, China, and the Ph.D. degree in EE from Xidian University, Xi'an, in 1994.

He is an Associate Professor with the Department of ECE, Michigan State University, East Lansing, MI, USA. His current research interests include network security, cloud computing security, privacy-preserving communications, distributed network storage, and Internet of Things.

Dr. Ren is a recipient of the U.S. National Science Foundation Faculty Early Career Development (CAREER) Award in 2009. He served as the TPC Chair of IEEE ICNC'17, the General Chair of ICNC'18, and an Executive Chair of ICNC'19 and ICNC'20. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE INTERNET OF THINGS JOURNAL, and the *ACM Transactions on Sensor Networks*, and a Senior Associate Editor for *IET Communications*.