Advances in Blockchain Security

Truc Nguyen, Tre' R. Jeter, and My T. Thai

Abstract Blockchain, the technology that underpins the great success of Bitcoin and various other cryptocurrencies, has incredibly emerged as a trending research topic in both academic institutes and industry associations in recent years. With great potential and benefits, the blockchain technology can stimulate a new decentralized platform for various applications such that the possibility of censorship, monopoly, and single point of failures can be eliminated. However, the blockchain is still in its early stages and not yet ready to realize that vision, since there are many security vulnerabilities that can be exploited to obstruct blockchain systems. In this chapter, we present fundamental challenges and recent advancements in the blockchain technology, especially in terms of security. In particular, we investigate the security threats of blockchain, effectively capturing the recent attacks, and review some security enhancement solutions for blockchain.

1 Introduction

Since the original paper in 2009 [1], Bitcoin has gained much attention from both academic institutes and industry associations. With a market capitalization of more than one hundred million dollars [2], Bitcoin is undoubtedly one of the most successful cryptocurrencies, averaging thousands of transactions per day. Blockchain is the technology that underpins the success of Bitcoin. At its core, blockchain is essentially a distributed ledger of transactions maintained by a set of nodes that do not trust one another. By using a consensus mechanism, nodes in a blockchain network agree on an ordered set of linked data blocks that each contains multiple valid and digitally signed transactions. The main selling point of blockchain is a decentralized nature

T. Nguyen (✉) · T. R. Jeter · M. T. Thai
University of Florida, Gainesville, USA
e-mail: truc.nguyen@ufl.edu

T. R. Jeter
e-mail: t.jeter@ufl.edu

M. T. Thai
e-mail: mythai@cise.ulf.edu

D. A. Tran et al. (eds.), *Handbook on Blockchain*, Springer Optimization and Its Applications 194, https://doi.org/10.1007/978-3-031-07535-3_11

in which applications can operate efficiently without the need of a central authority. From the perspective of database systems, blockchain can also be viewed as a distributed database for transaction management. While traditional databases assume a trusted environment, nodes in a blockchain network can behave in arbitrary manner.

One of the core components of blockchain systems is a consensus mechanism that is used to achieve verifiable decentralized consensus in the presence of malicious nodes. This is also referred to as making blockchain Byzantine Fault Tolerant, or BFT. A consensus mechanism can take the form of a probabilistic (e.g., Proof of Work/Proof of Stake) or deterministic (e.g., Practical BFT [3]) algorithm. Finality achieved via probabilistic consensus algorithms is temporary, nonetheless, as more blocks are added to the chain over time, the probability of overturning the previous blocks become smaller, approaching zero. By design, blockchain can tolerate Byzantine failure, thus it offers stronger security than conventional database systems.

Bitcoin, in its original design, is a blockchain that stores coins and is limited to facilitating financial transactions that move coins from one address to another. Since then, blockchain has evolved beyond cryptocurrencies to support any arbitrary, programmable transaction logic [4]. For example, Ethereum is a blockchain that enables any decentralized applications in the form of smart contracts. In the context of blockchain, *smart contracts* are defined as self-executing and self-enforcing programs that are stored on chain. They are intended to facilitate and verify the execution of terms and conditions of a contract within the blockchain system. By employing this technology, applications that previously require a trusted intermediary can now operate in a decentralized manner while achieving the same functionality and certainty. For that reason, blockchain and smart contracts together have inspired many decentralized applications and stimulated scientific research in diverse domains [5–9].

Unfortunately, due to its popularity and the value of cryptocurrencies, efforts have been made to exploit the weaknesses and vulnerabilities of blockchain. As a result, it is known to be susceptible to various security issues [2, 10] and was attacked multiple times in the last 10 years. For this reason, the blockchain technology is still in its early stage and not yet ready to realize its full potential.

In this chapter, we conduct a comprehensive survey on recent advances in blockchain that aims to make the technology more practical and deployable. In specific, we present some security threats of blockchain, especially the vulnerabilities in the blockchain network and smart contracts, which effectively capture past attacks to the blockchain. Then, we review the security enhancement solutions for blockchain and how they would affect the scalability and decentralization. Finally, we survey some other significant advances in blockchain including blockchain anonymity, consensus protocols, and the use of secure hardwares in blockchain.

Organization. The rest of the chapter is structured as follows. Section 2 establishes some background knowledge on cryptology and blockchain technology. Section 3 describes some recent security threats of blockchain, especially on the blockchain network and smart contracts and also shows some security enhancement solutions. In Sect. 4, we present other notable advances in blockchain in terms of privacy and consensus protocols. Finally, Sect. 5 concludes the chapter.

2 Background

This section covers necessary background knowledge for discussing security issues in blockchain. Specifically, we present some cryptographic primitives, including public-key cryptography and cryptographic hash functions, and a general explanation of blockchain technology and transitions from known cryptographic practices to more advanced and practical security methods used in blockchain.

2.1 Cryptographic Primitives

Cryptographic Hash Functions. A cryptographic hash function is generated by a mathematical function that compresses information in a string of letters and numbers of a fixed size. Cryptographic hashes are one-way functions. A one-way function is a function that can be computed easily and quickly for any input, but is very difficult to revert back to the original input [11]. A cryptographic hash function follows this same method. Any input can be "hashed", but the computational complexity to revert it back to the original input proves to be exceedingly difficult.

Cryptographic hash functions should be (1) pre-image resistant, (2) collision resistant, and (3) second pre-image resistant. Pre-image resistance directly corresponds to the one-way functionality of hashing functions. For any hash value, it should be very difficult and nearly impossible to read the corresponding message related to that hash. Denoting $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ as a public cryptographic hash function that maps a bitstring of arbitrary length to a bitstring of fixed-length l, the pre-image resistance property states that, given a hash value h, it is infeasible to find any message m such that $h = H(m)$. Formally speaking, for any probabilistic polynomial algorithm A_1, we have

$$\Pr[m \leftarrow A_1(1^\lambda, h) | h = H(m)] < negl(\lambda) \tag{1}$$

where $negl(\cdot)$ denote some negligible function, and λ is a security parameter.

A hash function that attains second pre-image resistant must be so complex that it is computationally difficult to find a second input message that will result in an identical hash output. Specifically, given a message m_1, second pre-image resistance makes it computationally infeasible to find a message $m_2 \neq m_1$ such that $H(m_1) = H(m_2)$. This property can be defined formally as follows

$$\Pr[m_2 \leftarrow A_2(1^\lambda, m_1) | m_1 \neq m_2 \wedge H(m_1) = H(m_2)] < negl(\lambda) \tag{2}$$

for any probabilistic polynomial algorithm A_2.

Collisions in hash functions refer to the chance that two inputs' hash values are equivalent to one another. A cryptographic hash function should be collision resistant in that it is difficult to find two distinct messages $m_1 \neq m_2$ such that $H(m_1) = H(m_2)$. In other words, it ensures that, for any probabilistic polynomial algorithm A_3, the following holds:

$$\Pr[(m_1, m_2) \leftarrow A_3(1^\lambda) | m_1 \neq m_2 \wedge H(m_1) = H(m_2)] < negl(\lambda) \tag{3}$$

Moreover, it can be shown that collision resistance implies second pre-image resistance. Suppose there exists a polynomial algorithm A_2 that can violate equation (2), an adversary can devise a polynomial algorithm A_3 as follows: pick a random message m_1 and obtain $m_2 \leftarrow A_2(1^\lambda, m_1)$ in polynomial time. This results in $m_2 \neq m_1$ and $H(m_1) = H(m_2)$, thus violating equation (3).

The "birthday paradox" places an upper bound on the computational difficulty of a collision-finding algorithm: if the output length of a hash function is l bits, then an attacker who computes hashes of $2^{l/2}$ random inputs can find a collision with probability greater than 0.5. A hash function is considered flawed if a collision can be found by a method easier than this brute-force attack.

Public Key Cryptography. Public key cryptography was the solution to two problems: key distribution and signatures [12]. Sometimes called asymmetric cryptography, this cryptographic system involves a pair of keys: a public key and a private key. The public key can be distributed to all users sending encrypted messages to one another and the private key is always kept private. When a message is sent over an insecure network, it is encrypted using the public key. The recipient will then use their private key to decrypt the message.

Public key cryptography is also used to authenticate users. A message can be combined with a user's private key to generate a digital signature on top of the message. Another user with the related public key can also combine the same message with a known signature. If the generated signature matches the message that was sent from the first user, then that user is said to be authenticated and trusted.[1] The message itself is also verified.

Asymmetric cryptographic algorithms are slower than symmetric cryptographic algorithms, but still useful. Some algorithms are built for key distribution and privacy such as the Diffie-Hellman key exchange. There are algorithms such as the Digital Signature Algorithm that only create digital signatures. However, when algorithms like the two above-mentioned are combined, the Rivest-Shamir-Adleman (RSA) algorithm is the result. This algorithm allows for users to openly share encrypted files, data, or other information through the internet or email for example. The public key encrypts the message and only the recipient's private key can decrypt the message [13]. RSA is widely used today for secure data transmission and an easy way to implement multi-factor authentication into secure systems.

[1] https://www.ibm.com/docs/en/ztpf/1.1.0.14?topic=concepts-digital-signatures.

Taking these algorithms further in application, implementing a Public Key Infrastructure (PKI) would further verify and authenticate users. A PKI is a system of a third-party user called a Certificate Authority (CA) that certifies the ownership of a set of keys. This system is good for avoiding attacks because it is a set of protocols that manage overall public-key encryption and the creation, distribution, use, storing, and disabling of digital certificates.

2.2 Blockchain Primer

Peer-To-Peer Network. A peer-to-peer network is a decentralized network that is maintained by a distributed group of users that can act as servers and clients. Unlike the traditional Client-Server network, a peer-to-peer network does not have a central server. By definition, a peer-to-peer network consists of each node/peer within that network providing and making their personal resources accessible to others on the network without the need for an intermediary entity like a central server [14]. The peer-to-peer network plays a huge role in blockchain technology because it allows for transactions of any kind without the need for a middle-man or central server. In this distributed network, any user can verify or validate transactions and be a part of the process of creating new blocks by simply setting up a node on the blockchain.

The decentralized nature of blockchain technology makes it easily accessible and available. The peer-to-peer make-up also ensures resiliency. If one peer goes down, the other peers within the network are not affected and can maintain work. Blockchain technology has consensus constraints as does a stand-alone peer-to-peer network which mitigates a blockchain from many malicious activities. This network is also nearly impossible to execute a Denial-of-Service attack because there is no central server to attack.

Blockchain Architecture. A blockchain is essentially a database of records (transactions) shared across a peer-to-peer network [15]. Because of its immutable nature, changing a block on the blockchain is very difficult to do without being noticed. A record on a blockchain can be any piece of information such as bank transactions, health information, purchases, voting results, cryptocurrency, etc. A block on the blockchain is made up of a group of records. The actual blockchain itself is all of the blocks linked to one another.

Each record lists the details of each transaction and appends a digital signature from all who were involved with the transaction. The record is then verified by the network by every participant (node) to check the validity of the transaction. This process is called consensus. A decentralized network of nodes must come to a consensus before a block is added to the blockchain by any of the consensus algorithms (i.e., PoW, PoS,PoA, PoAh, etc).[2] Once a record is verified by the network, it is added to a block.

[2] https://www.investopedia.com/terms/c/consensus-mechanism-cryptocurrency.asp.

Every block in a blockchain has a cryptographic hash as its unique identifier. Each block contains the hash of the previous block and its own unique hash. Because these hash values match with each subsequent block, it is very difficult (nearly impossible) to change information on the blockchain without being noticed. Altering a block will change the hash of that block and break the chain. If the hashes of the previous block and current block do not match, then the blockchain has been altered. To restore the altered block, one would have to recalculate the original hash and each hash of the following blocks.

Double-Spending and Reaching Consensus. The Bitcoin and blockchain technology in general were motivated by the double-spending attack. In digital currency systems, double spending is the act of successfully spending some coins more than once.

In conventional systems, all the transactions are validated and recorded in a centralized manner. However, in blockchain, every node processes transactions and keeps a copy of the ledger. Due to the fact that multiple copies of blockchain are stored at different nodes in the network, it is challenging to maintain a consistent global view of the blockchain in the whole system. In particular, a node can simultaneously issue two different transactions on the same set of coins as input, to two different receivers. If both the receivers successfully validate the transaction independently based on their local view of the blockchain, then their copies of the blockchain become different and the blockchain ends up with **forks**. Specifically, the nodes will have different views of the global state and the network will no longer be consistent unless we resolve this fork. This type of malicious behavior makes decentralized currency particularly vulnerable to double spending.

Therefore, a distributed consensus mechanism is needed in a blockchain network to tackle this issue [6]. Intuitively, all nodes in the blockchain could vote on the order of transactions for each block, and the result is decided by the majority. Unfortunately, in an open network where anyone can participate, this mechanism would not be secure due to the **Sybil attack**: a single entity can generate various identities, vote several times, and thereby becoming the majority of the network. In other words, any adversary can easily take over the blockchain.

Bitcoin tackles this issue by proposing the proof-of-work mechanism [1], where each node has to solve a computationally expensive puzzle to vote for a block. This is also referred to as *mining*. As a result, generating several sybil identities on the blockchain is futile, as the computing resources of any single node are limited. In the event of a fork, the proof-of-work mechanism ensures that the nodes choose the fork that contains the most amount of work, that is, choosing the longest fork. Hence, this enables the network of untrusted nodes to reach consensus on the proper order of transactions.

Bitcoin Mining. Mining is where cryptographic hashes come into play with Bitcoin and blockchain technology. Miners maintain the consistency and immutability of the blockchain by placing each new transaction into a new block to be placed on the blockchain. Miners send this new block to the entire network for validation and consensus before the block is added to the blockchain. Every block has a tag in the form of a SHA-256 cryptographic hash [16] of the previous block and its own cryptographic hash.

Blocks are only accepted if they contain a valid proof-of-work (PoW). The PoW is generated by double hashing a block header, which contains a nonce, using the SHA-256 function. To make a valid PoW of a block, the miners need to find a nonce so that the resulting hash output is smaller than the difficulty target determined by the network. The higher the target value, the easier it is to find the correct PoW combination.[3] A nonce is a number that is only used once and is usually a sequence of natural numbers. This process gives the blockchain its immutable nature. Depending on the difficulty target, it usually takes an immense amount of computing power to generate a valid PoW for a new block. Furthermore, because the block will have the cryptographic hash of the previous block and its own attached, the block is very difficult to alter without invalidating the chain. Altering the transaction information of the block will change the hash of the block which will render the block compromised.

Ethereum and Smart Contracts. Ethereum is another decentralized network similar to Bitcoin that was created in 2013 by Vitalik Buterin. Its cryptocurrency is called Ether and is only second in value to Bitcoin. Ethereum allows for the creation of non-fungible tokens (NFTs). These tokens can represent anything that has an actual value as a unique item like art, photos, or any digital files. The distributed ledger within blockchain is then used to verify ownership of these items creating a safe and efficient marketplace. The main difference between Ethereum and Bitcoin is Ethereum's use of smart contracts.

Smart contracts are programs that are stored and executed by all nodes in the Ethereum blockchain using the Ethereum Virtual Machine (EVM). The EVM is basically a stack-based virtual machine that supports a Turing-complete programming language. Smart contracts can be deployed and triggered by the blockchain transactions. Each operation on the EVM costs some amount of gas that determines the fee needed to execute the smart contract. The transaction is assigned with a bounded amount of gas, and when that amount is exceeded, the entire execution is terminated and the operations are reversed. In contrast to conventional programs, smart contracts are immutable by design. Therefore, programming mistakes or vulnerabilities on the smart contract cannot be reversed or fixed.

In Ethereum, a smart contract can utilize three memory regions to perform data operations during execution: stack, memory, and storage. A (data) stack is a virtual stack that can be used to store data. Note that EVM also has a call stack, which is different from the data stack. The memory is a byte-addressable region allocated at run-time. Storage is implemented using a key-value store. The stack and memory are both volatile, meaning that the data stored are cleared after each execution. However, the storage is persistent, which can be used to store data across transactions.

[3] https://medium.com/geekculture/the-implication-of-bitcoins-proof-of-work-algorithm-40921bb13530.

3 Blockchain Security: Attacks and Counter-measures

This section presents a comprehensive survey on some security issues of blockchain, especially focusing on attacks and threats on the blockchain network and smart contract.

3.1 Blockchain Network

Attacks on Peer-to-Peer Network. By design, the blockchain peer-to-peer network is open, decentralized, and independent of a public-key infrastructure. Hence, it does not employ cryptographic authentication between nodes, and they are identified purely by IP addresses. In the Bitcoin network, each node is implemented to use a randomized protocol to select eight peers to create outgoing connections. Nodes with public IPs accept up to 117 incoming connections from any IP addresses. Nodes exchange their local views of the state of the blockchain with their connected peers.

However, this open nature of blockchain makes it feasible for adversaries to join and attack the peer-to-peer network. Heilman et al. [17] investigate an eclipse attack on the bitcoin network where the attacker takes control over all of the victim's incoming and outgoing connections, thereby isolating the victim from the rest of its peers in the network. After that, the attacker is free to manipulate the victims' view of the blockchain, force the victim to waste computing power on obsolete views of the blockchain, or exploit the victims' mining power for its own malicious purposes. The authors present an off-path attack in which the attacker only controls endhosts but not key network infrastructure between the victim and the rest of the Bitcoin network. The attack mainly forms incoming connections to the victim from a set of controlled endhosts, sends fake network information, and waits until the victim restarts. With high probability, the victim then creates all eight of its outgoing connections to attacker-controlled endhosts. Additionally, the attacker also monopolizes the victims' 117 incoming connections.

Some security implications of this attack include: (1) An attacker can hoard blocks discovered by eclipsed miners, and release blocks to both the eclipsed and non-eclipsed miners once a competing block has been found, thereby making the eclipsed miners waste computing resources on orphan blocks; (2) selfish mining [18]; (3) eclipsing miners eliminates their mining power from the rest of the network, making it easier to for the attacker to becomes the majority in the network; (4) double spending [19].

On the other hand, Apostolaki et al. [20] exploit the Bitcoin hosting centralization issue to conduct a routing attack. Although one can run a Bitcoin node, the nodes that form the Bitcoin network today are far from being distributed uniformly around the globe. Specifically, their experimental results illustrate that few Internet Service providers (ISPs) host most of the Bitcoin nodes. Specifically, 13 ISPs, which is about 0.026% of all ISPs, host roughly 30% of the entire Bitcoin network. Furthermore, a majority of the network traffic between Bitcoin nodes propagate over only a few ISPs. Indeed, their experiment shows that 60% of all possible Bitcoin connections cross three ISPs. Simply speaking, three ISPs observe, drop, or modify 60% of all Bitcoin traffic.

The authors [20] present how an adversary can utilize the network infrastructure to perform (1) an eclipse attack and (2) a delay attack. The eclipse attack works by intercepting network traffic between blockchain nodes. To do so, an attacker can leverage the fact that Border Gateway Protocol (BGP), an Internet routing protocol, does not verify the source of routing announcements. This attack involves getting a router to spread false announcements that it has a shorter path to certain IP prefixes, thereby maliciously rerouting Internet traffic. From that, the attacker can proceed with hijacking all the IP prefixes associated with the nodes in one component, effectively intercepting all the network traffic exchanged between that component and the rest of the network. This is commonly referred to as a BGP hijacking attack. Once the path is hijacked, the attacker can drop all these connections to disconnect that component from the rest of the network, thus eclipsing the miners. The network centralization of Bitcoin nodes as above-mentioned further exacerbates the issue as few IP prefixes need to be hijacked, making eclipse attacks particularly feasible. In fact, their study shows that 39 prefixes, accounting for 0.007% of all Internet prefixes, host 50% of Bitcoin mining power. This implies that, by hijacking only those 39 prefixes, an attacker is able to isolate roughly 50% of the mining power. BGP hijacking attacks that involve orders of magnitude more IP prefixes are routinely seen in the Internet today.

Besides eclipse attack, a delay attack can be conducted based on the fact that Bitcoin nodes are implemented to send a request for blocks to only one peer to prevent the network from being overwhelmed with the transmissions of blocks. If the peer is not responsive for 20 minutes, an alternative peer will be selected to send the request to. This implementation, together with the fact that Bitcoin messages are exchanged in plaintexts, allows for an effective attack where attackers try to prolong block transmissions by delaying or dropping those requests for blocks. Specifically, the attacker can simply modify to the content of the Bitcoin messages that they intercept. Since the Bitcoin protocol does not offer protection for those messages, both the receiver and the sender become oblivious of the fact that the message has been tampered with, thus enabling a very stealthy attack. The implication is that the attacker can then conduct other attacks like double spending or try to waste computing power of honest miners. What makes such delay attacks feasible and practical is the centralization of Bitcoin nodes in a small number of networks and prefixes, as well as the centralization of mining power in some certain mining pools. The authors discover that three ISPs control a majority of all Bitcoin traffic. This implies that these ISPs can stealthily interfere with Bitcoin traffic. In contrast to eclipse attacks, delay attacks could not disrupt the whole blockchain system, but rather reduce the performance of the network. Thus, even if many nodes are slowed down under attack, the Bitcoin system would still be able to function, but at a lower performance and less secure.

Saad et al. [21] propose some potential attacks based on spatial and temporal characteristics of the Bitcoin network. They investigate three different levels of attacks, emphasizing the network centralization. At the network level, due to the increasing centralization of the Bitcoin network, the authors are able to empirically demonstrate that an attacker can easily partition the network spatially through BGP hijacking by controlling only a few ASes, thus causing a hard fork. At the AS level, they discover that in certain cases, by hijacking roughly 20 prefixes, the adversary can gain control over more than 80% of the Bitcoin nodes that are placed inside the same AS. At the organization level, they show that multiple ISPs control more than one AS, which results in even more centralization, and facilitating new attack avenues. Furthermore, they leverage the non-uniform consensus among connected nodes to propose temporal attacks. They observe that there is a significant delay in consensus and block propagation because of the latency and adversarial peer behavior. Their study suggests that even after a few minutes from the publication of a block, about 62.7% of nodes in the network are not up-to-date and still remain behind the latest block by one or two blocks. As a result, it is suggested that such a behavior can be leveraged to optimize an attack where false blocks are fed to nodes, thereby temporally partitioning the network.

Since those above-mentioned attacks are based on BGP hijacking, however, due to the openness of BGP operations, such a hijacking attempt can be observed globally, thereby enabling instant attack detection and attacker identification. Specifically, the real identity of the attacker (i.e., the malicious AS) is instantly revealed to the public. As such, this can be a deal-breaker for large ASes since attempting the attack can potentially damage their reputation. Tran et al. [22] present a more stealthy Bitcoin attack, which is referred to as EREBUS, that enables a network attacker to control the peer connections of a victim Bitcoin node without manipulating the network routing protocol, thereby eliminating control-plane evidence of attacks. This is possible because the attack strategy only exploits data-plane attack messages, so it remains invisible to any control-plane monitoring systems. Furthermore, even if data-plane traces of the attack are detected, the attack still offers plausible deniability. The authors demonstrate that Tier-1 or large Tier-2 ISPs can conduct this attack to target a majority of thousands of Bitcoin nodes in the system that accept incoming connections from other nodes. Consequently, attackers who control large ISPs (such as nation-state adversary), are capable of launching the EREBUS attack stealthily.

At a high level, EREBUS works as follows. Without interfering with the underlying routing protocols, the adversary AS alters the existing outgoing peering connections of a victim node to the new connections with the Bitcoin nodes whose victim-to-node inter-domain paths include the adversary AS. Eventually, the malicious AS will be placed on the paths of all the peer-to-peer connections of the victim node. The attack is feasible not because of the implementation of Bitcoin nodes but the inherent topological advantage of being a network adversary. In specific, as a man-in-the-middle adversary, the EREBUS malicious AS can exploit an enormous amount of network addresses reliably over a long period of time.

Counter-measures. Two counter-measures are typically recommended for this type of network attack: (1) disable incoming connections and (2) only make outgoing connections to well-connected or known/whitelisted miners. However, there are several problems with scaling this to the full Bitcoin network. First, if incoming connections are disabled on all current nodes, how do new nodes join the network? Second, how does one decide which peers to connect to? Who determines the whitelist of miners? In [17], the authors propose a set of counter-measures that partially preserve openness by allowing unsolicited incoming connections, while raising the threshold for eclipse attacks. The counter-measures ensure that, with high probability, if a victim stores enough legitimate miners that accept incoming connections, then the victim cannot be eclipsed regardless of how many IP addresses the attacker controls.

In [23], the authors propose the SABRE network to secure Bitcoin against the BGP hijacking attacks. SABRE is a Bitcoin relay network that relays blocks worldwide through a set of connections that are resilient to routing attacks. SABRE is designed to be secure and scalable and is able to run alongside the existing peer-to-peer network and can be deployed easily. SABRE is specifically designed to protect both relay-to-relay and relay-to-client connections. At a high level, to secure relay-to-relay connections, SABRE places nodes in ISPs that connect directly to one another, creating a fully connected graph of direct links and also in /24 prefixes. To secure relay-to-client connections, relay nodes are placed in a way that most nodes have for each potential attacker at least one route to SABRE that is more preferable than any route that this attacker can advertise, thereby tackling the BGP hijacking attacks. The main technical insight is that SABRE leverages fundamental properties of BGP policies to host relay nodes in networks that are essentially protected against routing attacks, and on network routes that are preferable by the majority of Bitcoin nodes. These properties are generic and can be used to protect other blockchain networks. However, this approach introduces a trusted entity to the system to control the network connections between nodes, which violates the trust model of blockchain.

The authors in [22] propose a set of counter-measures to defend against stealth BGP hijacking attacks. First, some third-party proxies can be used to verify the reachability of IP addresses. However, this approach has limited scalability because creating multiple proxies at different locations for thousands of potentially vulnerable nodes in the Bitcoin network would be difficult in practice. Furthermore, because of the limited scalability, any proxy-based approaches could eventually result in few centralized proxies. Another solution is increasing the amount of outgoing connections that a Bitcoin node is able to make. According to the authors, the increase can in fact potentially sabotage the network if it is not deployed properly. This is because it may instantly boost the amount of network connections and the volume of network traffic in the system. This sudden increase can potentially exacerbate the delay of transactions and blocks in the system. Therefore, the practicality of these counter-measures remains questionable.

3.2 Smart Contracts

Ensuring the correctness of smart contracts is a critical and urgent security concern. Nowadays, billions of dollars are handled by smart contracts, and only in the past couple of years, millions of these have been lost by adversaries who exploited subtle flaws in the logic of the contracts [24, 25]. In fact, Ethereum already encountered a lot of disastrous attacks on vulnerable smart contracts. The most notable ones are the DAO hack in 2016[4] and the Parity Wallet hack in 2017,[5] together resulting in a loss of over 300 million US dollars. The problem is exacerbated as the smart contracts become immutable once placed on the blockchain, hence bugs and flaws found after deployment cannot be fixed.

Real-World Vulnerabilities of Smart Contracts. Below is a list of vulnerabilities in Ethereum smart contracts according to [7].

- *Airdrop hunting.* Airdrop is a method to reward new users a small amount of tokens as a way of promoting attention and appealing to more users. Airdrop hunting is an attack strategy that leverages the weaknesses of airdrop and bypasses the identity verification of new users to keep generating new sybil users to obtain a large amount of free tokens.
- *Call injection.* Call injection is a method that allows any contract to call any function in a vulnerable contract. It is often used to modify ownership and trigger money transfers.
- *Reentrancy.* A reentrancy attack happens a function is created that makes an external call to another untrusted contract before it updates its own state. A reentrancy attack may lead to a repeated transfer of money from the victim to the adversary, thereby exhausting the balance of the victim contract.
- *Honeypot.* A honeypot is a bait that lures a victim into losing tokens.
- *Call-after-destruct.* Call-after-destruct is the act of calling a function in a destructed contract with tokens, resulting in the loss of these tokens.

Common Attacks on Smart Contracts. In [7], the authors conduct an analysis of real-world attacks based on the log of transactions generated by "uninstrumented" Ethereum Virtual Machine (EVM). In specific, they capture two essential behaviors of a malicious transaction: (1) it attempts to exploit a vulnerable contract and (2) it often results in ether or token transfers. The results unveil a large volume of attacks that is greater than what have been discovered in the literature. In particular, airdrop hunting and zero-day variants of known vulnerabilities are often the targets of those attacks.

One of the most common attacks is luring victims into traps. This type is also commonly referred to as *honeypot*, as it often involves setting up a bait to attract victims. Honeypots are smart contracts that seem to have some apparent flaws and

[4] https://www.coindesk.com/understanding-dao-hack-journalists.

[5] https://medium.com/@Pr0Ger/another-parity-wallet-hack-explained-847ca46a2e1c.

bugs in their design and implementation. For instance, several Ethereum smart contracts enable any malicious user to retrieve ether (Ethereum's cryptocurrency) from the contract's balance, given that the user previously transfers a certain amount of ether to the contracts in the first place. However, once the user tries to take advantage of this obvious vulnerability, a second trapdoor (unknown to the user) opens and prevents the draining of ether from succeeding. The key observation here is that the user only pays attention to the obvious flaw and does not think of the possibility that some other vulnerabilities might be concealed within the smart contract. In the same manner as other types of fraud, honeypots exploit the fact that human beings are usually greedy and easily manipulated.

In [25], the authors investigate the incidents of such honeypot smart contracts in Ethereum and introduce HONEYBADGER, a toolbox that uses a combination of symbolic execution and precise heuristics to automatically detect various types of honeypots. By using HONEYBADGER, users have the capability of providing interesting insights on some properties of honeypots that are being hidden in smart contracts on the Ethereum blockchain.

Another attack on Ethereum smart contracts is to exploit several flaws in the *metering* mechanism of Ethereum to conduct a DoS attack. This metering mechanism is used to assign a gas cost to smart-contract execution in order to incentivize miners to operate the blockchain system and protect it against DoS attacks. In the past, several problems in the implementation of Ethereum metering mechanism allowed several DoS attacks.

In [26], the authors unveil a number of issues in the Ethereum metering model, especially some substantial discrepancies in the pricing of the Ethereum instructions. Additionally, they found that the correlation between the gas cost and the utilized computing resources, such as CPU and memory consumption, is very small. To conduct this study, they use a large amount of Ethereum smart contracts to determine some critical edge cases that point out several problems in EVM metering. First, there are several EVM instructions that cost significantly less gas than their actual resource consumption. Second, there are cases where the cache substantially impacts the execution time.

From these findings, the authors present a new DoS attack called Resource Exhaustion Attack targeting Ethereum smart contracts, which uses these flaws to generate low-performance contracts in terms of throughput. The challenging part is how to produce well-formed EVM contracts that minimize the throughput. The proposed attack combines empirical data and a genetic algorithm so as to create low-throughput contracts on Ethereum. As a result, the authors are able to generate contracts that are about a hundred times slower in average than typical contracts. They also show that most current Ethereum client implementations are vulnerable to this attack and those clients would not be able to stay in sync with the rest of the network when under attack. The authors have disclosed this vulnerability to the Ethereum Foundation and were awarded 5,000 USD [26].

Formal Verification of Smart Contracts. Researchers believe that smart contracts, similarly to any safety-critical system, must be formally verified before deployment

[8, 24, 27]. ZEUS [8] is a practical framework for automatic formal verification of smart contracts using abstract interpretation and symbolic model checking. At a high-level view, ZEUS works as follows. With smart contracts that are programmed in high-level languages, ZEUS leverages user assistance to formulate the criteria relating to correctness and fairness. These contracts and the policy specification are then translated into a low-level intermediate representation (IR) that encodes the execution semantics to properly inspect the behavior of the contract. After that, static analysis is performed based on the IR to identify the points at which the verification predicates (as defined in the policy) must be asserted. Finally, the modified IR is fed to a verification engine that ensures the safety of the smart contract.

In [24], the authors list out two crucial challenges of building an automated verifier for smart contracts. First, via function calls, smart contracts that we want to verify usually communicate with some external contracts. Consequently, as we do not know the code of the external contracts, these external contracts may eventually trigger the original contract in some arbitrary ways. It is very challenging to devise automated verification when there are potentially a large number of arbitrary callbacks from unknown external contracts. Second, the number of transactions that smart contracts process is unbounded. Considering processing a single transaction as an iteration in a loop, the functions in smart contracts are indeed implicitly executed in an infinite loop. Thus, even though smart contracts often do not have loops, the verifier still needs to soundly handle loops.

To address those challenges, the authors in [24] propose VERX, an automated verifier of functional requirements for Ethereum smart contracts. VERX is mainly motivated by the practical challenges that emerge when assessing real-world smart contracts. One of the main insights is that most practical contracts use a defensive strategy against external callbacks by making sure that these do not create any new behaviors. Specifically, any behavior with external callbacks is considered as another behavior without external callbacks; these are referred to as external callback free (EECF) contracts. VERX focuses on verifying EECF contracts as they offer two essential benefits. First, formalization of requirements is simplified, as auditors can write the specification without explicitly considering all possible external callbacks. Second, exploring all possible external callbacks is not necessary, thereby enabling precise and scalable analysis.

Ensuring Privacy of Smart Contracts. When implementing applications in smart contracts, one of the major concerns is data privacy. Since smart-contract transactions are processed by the blockchain's nodes, transaction data have to be made available to all nodes. Hence, it is not trivial to preserve data privacy on smart contracts without violating the security model of blockchain. This is a major problem for applications that deal with sensitive data such as voting or healthcare applications.

Most approaches to enforcing privacy use cryptographic protocols to both secure secret data and validate the integrity of computations on blockchains like Ethereum without altering their trust model. In particular, Non-Interactive Zero-Knowledge (NIZK) proofs allow a prover to prove statements involving private data without revealing any information other than the correctness of the statements. NIZK basically satisfies four properties: (1) completeness (if the statement is correct, the probability that an honest verifier accepting the proof from an honest prover is 1); (2) soundness (if the statement is incorrect, with a probability less than some small soundness error, an honest verifier can accept the proof from a dishonest prover showing that the statement is correct); (3) zero-knowledge (during the execution of the ZKP protocol, the verifier cannot learn anything other than the fact that the statement is correct); and (4) non-interactive. Practical NIZK proof constructions have been proposed and made available in Ethereum.

The paper [9] presents Hawk, a decentralized smart-contract system that does not store blockchain transactions in the clear on the blockchain, thereby preserving data privacy for the transactions, effectively concealing them from the public view. The main advantage of Hawk is that a smart-contract developer can program a private smart contract in a simple manner without having to develop any cryptographic schemes. Then, the Hawk compiler will generate an efficient cryptographic protocol in which contractual parties interact with the blockchain, using cryptographic primitives such as NIZK proofs.

Another approach to a decentralized smart-contract system is the *zkay* language proposed in [27]. The authors introduce privacy types that define owners of private values. Zkay contracts are statically type checked to ensure they are realizable using NIZK proofs and to prevent unexpected information leakage. To enforce zkay contracts, the compiler automatically converts them into contracts that have the same functionalities, retain the same privacy properties, and are executable on Ethereum.

3.3 Other Security Issues

Denial-of-Service. By design, blockchain platforms are appealing victims for Denial-of-Service (DoS) attacks: the rivalry among cryptocurrencies is very intense, and there are potential gains from short selling [28]. However, in practice, DoS attacks receive less attention comparing to other types of attack. This is due to the fact that traditional, network-based DoS attacks cannot scale to large decentralized systems, and that known DoS attacks on the mining process [29] are enormously costly. Specifically, mining-based DoS attacks require that the attacker's computing resources need to be greater than those of other miners combined, which is not practical.

Mirkin et al. [28] propose a Blockchain Denial of Service (BDoS) sabotage attack that is based on incentives: the underlying mechanism of the blockchain platform is targeted and the attacker tries to violate its incentive compatibility. In specific, the adversary uses its computing resources in order to convince honest miners to stop mining. In other words, the attacker can cause a blockchain system to stop its normal operation with only a fraction of other miners' resources. The key main insight in conducting this attack is that an attacker can manipulate the miners into thinking that the system is in a state that diminishes their revenue. The attack leverages the fact that the adversary can generate a block and broadcast only the block header as a proof to show that they mined it. The purpose is to show that they have an advantage over other miners, but do not have to reveal the block's content. The profit of a honest miner may decrease if they are oblivious of the block header, and thus they would be willing to receive the block headers. Therefore, miners are motivated to accept block headers. Simply ignoring the block header is not an effective defense strategy, since a miner is encouraged to receive block headers to maximize their payoff, such a defense strategy will not be employed by the miners.

In detail, the attack works in the following manner. The adversary generates a block B and broadcasts only the header of B. A miner may disregard the header of B and create a block following its previous block in the current chain, resulting in an additional branch of blockchain. Next, the adversary publishes the contents of B, resulting in two forks. Depending on the parameters and the state of the system, the miner's block may or may not be added to the main chain. The main idea is that when the expected profitability of the honest miners decreases, suppose that it is lower than some threshold, it is better for them to stop the mining process. If the decrease in profitability is substantial enough so that all miners decide to pause the mining process, the adversary can also stop mining. As a result, the blockchain mining comes to a complete halt, and new transactions will not be processed.

Mining Pool. Mining pools are formed by miners with the purpose of increasing the computing resource which may shorten the mining time of a block. Thus, it boosts the probability of obtaining the mining reward. Motivated by this benefit, a large number of mining pools have been formed in recent years, and many different mining strategies have been devised. In general, mining pools are managed by pool managers that forward unsolved work units to its members. The members are essentially miners of the Bitcoin network who decide to join a pool. Once a member mines a new block, the miner submits the block and the full proofs-of-work (FPoWs) to the manager. The manager sends the block to the Bitcoin network so as to obtain the mining reward. The reward is then distributed by the manager to participating miners based on how much they contribute to solving the mining puzzle. In specific, participants are rewarded based on the partial proofs-of-work (PPoWs) submitted to the manager. There are some open pools that allow participation from any miners, and private pools that only allow some authorized miners [2].

Due to the financial benefits of mining pool, the attack vector that targets the vulnerabilities in mining pool has been explored. Etay et al. [18] propose a selfish mining strategy to abuse Bitcoin's forks mechanism to obtain an unfair reward. Recall that only one branch of a fork can be accepted and others will be invalidated. In selfish mining, an attacker as a pool does not broadcast a block immediately, but instead builds a private chain internally. When the length of the public chain approaches its private chain, the attacker broadcasts the private chain, forcing other miners to accept this longer chain. Since the mining pool has large computing power, the attacker can earn a greater reward by invalidating blocks of honest miners, this also makes honest miners waste their computing resources.

Block Withholding (BWH) Attack. Different from the selfish mining, this attack is considered as an internal attack inside a mining pool. In this BWH attack, a malicious miner shares with the pool manager only PPoWs and keeps all the computed FPoWs to herself [30]. The pool manager is unaware of the blocks that were withheld and thinks that the attacker is still trying to use her computing resources to mine the block like other miners. The pool, being oblivious of this malicious behavior of the attacker, distributes its mining reward to her. Therefore, the malicious miner earns rewards without contributing anything useful to the pool. This is at the expense of the honest miners of the pool. On June 13, 2014, it was reported that a large-scale Block Withholding Attack attack was launched against Eligius, a popular mining pool, resulting in a loss of 5 million US dollar at the expense of honest miners.[6]

The authors in [30] propose a "sponsored block withholding attack". It can be observed that by conducting a BWH attack on a victim pool, the attacker indirectly increases the probability of wining the mining process for another pool. Thus, she can collude with some other pools to use a portion of her computing resources for attacking one pool and diminish the victim pool's chance of winning. In that scenario, she can be rewarded by the malicious pool for targeting the victim pool. The amount of reward can be determined according to the increase of profit to the malicious pool resulted from attacking the victim pool.

Kwon et al. [31] describe another attack called a fork after withholding (FAW) attack, which combines a BWH attack with intentional forks. In the same manner as the BWH attack, the FAW attack is always profitable regardless of an attacker's computing resources. In addition, the FAW attack provides much more rewards compared to the BWH attack. Particularly, the BWH attacker's reward is only the lower bound of the FAW attacker's. The authors propose two scenarios for this attack: single-pool and multi-pool.

In a single-pool FAW attack, in the same manner as a BWH attacker, an FAW attacker participates in the target pool and conducts an FAW attack against it. FPoWs are submitted to the pool manager by the attacker only when there is another miner who is not in the same pool submits a block. If the pool manager accepts the submitted FPoW and broadcasts the block, then a fork will be created. Because of the forks, all Bitcoin network participants will agree on only one branch. If the attacker's block is selected, the target pool will receive the mining reward, and thus, the pool will also reward her as well. In any case, the attacker is entitled to the extra rewards. The lower bound of the extra reward is the same for a BWH attacker.

On the other hand, to increase the reward, the attacker can conduct a multi-pool attack by simultaneously attacking n pools. The analysis shows that, as in the single-pool case, the FAW attack is always profitable, and the reward for an FAW attacker is greater than that for a BWH attacker. If the attacker executes the FAW attack against four currently popular pools, she will earn roughly 56% more reward than a BWH attacker does.

[6] https://bitcointalk.org/?topic=441465.msg7282674.

4 Other Significant Advances in Blockchain

Besides research efforts in preventing certain types of attacks on the blockchain network and smart contracts, this section shows some other notable advances in blockchain. Particularly, we focus on the privacy of blockchain transactions, consensus protocols, and the use of secure hardware in blockchain.

4.1 Anonymous Transactions

Most of anonymity vulnerabilities in blockchain arise because of the fact that Bitcoin, and many other blockchain platforms, associate each user with a pseudonym, and these pseudonyms are linked to financial transactions issued to the public blockchain. If an attacker can identify the user behind a pseudonym, the attacker may learn the user's transaction history. In practice, there are several ways to associate a user with her Bitcoin pseudonym. The most common method is to analyze transaction patterns in the public blockchain, and link those patterns using external information [32, 33].

Fanti et al. [34] investigate a lower-layer vulnerability: the networking stack. Whenever a user issues a transaction sending coins to another user, she first creates a transaction that contains the sender's pseudonym, receiver's pseudonym, and the transaction amount. This transaction is then broadcasted over the peer-to-peer network, which allows other users to validate her transaction and include it in the global chain. The authors demonstrate that, by using simple estimators to infer the source IP of each transaction broadcast, an eavesdropper adversary can link IP addresses to Bitcoin pseudonyms with an accuracy of up to 30%.

To address the anonymity issue in blockchain, Ben-Sasson et al. [35] propose Zerocash, a decentralized anonymous payments scheme for Bitcoin, that leverages recent advances in zero-knowledge Succinct Non-interactive ARguments of Knowledge (zk-SNARKs) [36]. The proposed payment scheme enables users to directly pay each other in a private manner: the transaction does not reveal the payment's origin, destination, and transferred amount. Zerocash extends and upgrades the Bitcoin protocol and software with anonymous transactions supporting privacy-preserving payments. As a result, despite using some of the same technology and software as Bitcoin, Zerocash becomes a new system that is distinct from Bitcoin. This new protocol introduces two types of coins: zerocoins (anonymous coins), and basecoins (non-anonymous coins). Comparing to Bitcoin's transactions, payment transactions created by the Zerocash protocol conceal any information that can be used to infer payment's origin, destination, or amount. Furthermore, the validity of the transaction can be verified on constant time via the use of a zk-SNARK. Users can convert from basecoins to zerocoins, send zerocoins to other users, and split or merge zerocoins they own in any way that preserves the total value, just as it is with Bitcoin.

However, it is worth noting that anonymous transactions take away the traceability of blockchain transactions. Basically, without knowing a transaction's origin and destination, it is impossible to trace back the transaction history. Some applications like supply-chain require a high degree of traceability, which means the transactions cannot be anonymous. The anonymity is also criticized for limiting accountability, regulation, and oversight. However, by using zk-SNARK, Zerocash is not limited to enforcing only the basic monetary invariants of a currency system. A wide range of policies can be supported by the underlying zk-SNARK cryptographic proof protocol. For instance, a user can prove in zero-knowledge that he paid his due taxes on all transactions without revealing those transactions, their amounts, or even the amount of taxes paid. In principle, if the policy can be specified by NP statements, it can be implemented using zk-SNARKs, and included in Zerocash.[7]

4.2 Consensus Protocols

Gilad et al. [37] present Algorand, a new consensus protocol that is designed to confirm transactions as fast as one minute. The core of Algorand uses a Byzantine agreement protocol, called BA, that scales to a large number of users, thereby allowing nodes in Algorand to agree on a new block in a short amount of time and without the possibility of forks. Algorand decides to employ BA due to the fact that it uses of verifiable random functions (VRFs) to randomly select users in a private, verifiable, and non-interactive way. Algorand mainly tackles three challenges: (1) it must avoid Sybil attacks, (2) it should scale to millions of users, and (3) it must be resilient to DoS attacks, and robust to users dropping out.

Algorand addresses these challenges in the following manner. First, Algorand assigns a weight to each user to prevent Sybil attacks. BA is designed to ensure consensus as long as a weighted fraction of the users are honest. Second, BA improves scalability by choosing a small committee that is formed by randomly selecting from the total set of users, to run each step in the protocol. All other users observe the protocol messages that allow them to learn the block that was agreed upon. Third, to hinder an adversary from manipulating committee selection, they are selected in a private, verifiable, and non-interactive way by the BA. In specific, each user in the system can independently and reliably determine whether they are chosen as a committee member, by computing a VRF that takes as input their private key and some information from the blockchain. Finally, to hinder an adversary from targeting a committee member after that member sends a message, BA requires committee members to speak only once. Therefore, once a committee member sends his message, hence revealing his identity to the adversary, the BA discards any further messages coming from that committee member.

In [38], the authors present Bitcoin-NG, a scalable blockchain protocol, that uses the same trust model as Bitcoin. Bitcoin-NG's latency and throughput are limited only by the propagation delay of the network and the processing capacity of the individual Bitcoin nodes, respectively. The key idea in designing Bitcoin-NG is decoupling

[7] http://zerocash-project.org/q_and_a.

Bitcoin's blockchain operation into two planes: leader election and transaction serialization. In particular, time is divided into epochs, where each epoch has a single leader. In the same manner as Bitcoin, a leader is elected randomly and infrequently. Once a leader is chosen, the leader is able to to serialize transactions at his or her discretion until the election of a new leader, which marks the end of the former's epoch. While this approach is substantially different from that of Bitcoin, the authors claim that Bitcoin-NG still maintains Bitcoin's security properties. In fact, leader election is already taking place in Bitcoin, though it is implicit. However, in Bitcoin, the task of the leader is serializing history, thereby freezing the system during the time between leader elections. On the contrary, leader election in Bitcoin-NG is forward-looking and ensures that the system is still able to process incoming transactions continuously.

Miller et al. [39] present an alternative to the Practical BFT [3] protocol, called HoneyBadgerBFT, the first practical asynchronous BFT protocol, which ensures liveness without making any timing assumptions. The authors make major efficiency improvements on the best state-of-the-art asynchronous atomic broadcast protocol that requires each node to transmit $O(N^2)$ bits for each committed transaction, thereby significantly limiting its throughput for all but the smallest networks. The cause of this efficiency is twofold. First, there is redundant work among the parties. However, naively eliminating the redundancy negatively impacts the fairness property, and paves the way for targeted censorship attacks. A solution is invented to overcome this problem by using a threshold public-key encryption scheme to tolerate these attacks. The second cause of the efficiency is the use of a suboptimal instantiation of the Asynchronous Common Subset (ACS) subcomponent. The authors show how to efficiently instantiate ACS by combining existing but overlooked techniques: (1) employ erasure codes for an efficient and reliable broadcast and (2) reduce ACS to reliable broadcast in the context of multi-party computation.

4.3 *Trusted Execution Environments (TEE) in Blockchain*

TEE in a computer system is realized as a module that performs some verifiable executions in such a way that no other applications, even the OS, can interfere [40]. Simply speaking, a TEE module is a trusted component within an untrusted system. Memory regions in TEE are transparently encrypted and integrity-protected with keys that are only available to the processor. TEE's memory is also isolated by the CPU hardware from the rest of the host's system, including high-privilege system software. Thus, this isolation protects the integrity and confidentiality of the enclave's execution from any malicious software running on the same system and ensures that the operating system, hypervisor, and other users cannot access the TEE's memory. Among available implementations of TEE, Intel SGX [41] supports generating remote attestations that are used to prove the correct execution of programs running inside TEE.

The authors in [42] offer a key observation that TEEs and blockchains have complementary properties. On the one hand, a blockchain can guarantee strong availability and persistence of its state, whereas a TEE cannot guarantee availability, since the host can arbitrarily terminate TEEs. Additionally, it cannot reliably access the network or persistent storage. On the other hand, a blockchain requires a huge amount of computing power, and exposes its entire state for public verification, while computation in TEE only incurs negligible overhead compared with native computation. TEE also offers verifiable computation with confidential state via remote attestation (e.g., SGX). Thus it is intuitive to build hybrid protocols that combine the advantages of both, in a way that we can exploit the immutability of blockchain to overcome the shortcomings of TEEs, and offload on-chain computation to TEE. However, note that using TEE also introduces a trusted entity to the blockchain system, which alters the trust model.

Cheng et al. [42] propose Ekiden, a system for highly performant and privacy-preserving smart contracts. The key idea behind the design of Ekiden is a secure and principled combination of blockchains and trusted hardware. Ekiden combines any desired underlying blockchain system with TEE-based execution. The design uses an architecture in which computation and consensus are separated. There are two main entities in the Ekiden architecture: compute nodes and consensus nodes. Compute nodes in Ekiden are tasked with performing smart-contract computation over private data off-chain in TEEs, then attesting the integrity of their execution on chain. In addition, the consensus nodes in Ekiden maintain the underlying blockchain, which do not need to use trusted hardware. Ekiden can be applied on top of any consensus mechanisms, in fact, it only requires a blockchain that can validate remote attestations from compute nodes. Therefore, the main advantage of Ekiden is that it can scale consensus and compute nodes independently according to performance and security needs.

In [43], the authors use TEE to improve the privacy of Bitcoin lightweight clients, in terms of concealing clients' addresses and transactions, without compromising the performance of the assisting full nodes. Specifically, they propose BITE, a solution in which an SGX enclave is run within an untrusted full node. The SGX enclave is tasked with validating transactions sent by clients. Since SGX provides code integrity and data confidentiality for enclaves, such a solution can preserve privacy and integrity of client requests. However, the authors also show that, although SGX can prevent a malicious software from directly accessing the enclave's memory, certain secret-dependent access patterns to external storage can still reveal the client's address. An example of such external storage is the transaction database. SGX is also susceptible to side-channel attacks, in which secret-dependent enclave data access patterns or control flow can be inferred by malicious software running in the same host. In specific, the adversary can monitor shared resources, such as caches, to gain insight into the execution of an SGX enclave. Taking into consideration such limitations of SGX, the authors devise a solution based on primitives such as oblivious transfer mechanisms, that enables client requests to be processed privately, even in the presence of the enclave's privacy leakage, without compromising the system's overall performance.

Lind et al. [44] leverage TEE to address the availability problem of state channels. The main insight is that, rather than having the parties to rely on the blockchain system to detect dishonest behaviors during off-chain transactions, they propose a design for a payment network in which parties use TEEs as a trusted entity to ensure correct protocol execution. In particular, they propose Teechain, a new payment network that supports highly secure and instant payments on existing blockchains. The main advantage of Teechain is that it only requires asynchronous blockchain access, that is, it makes no assumption on the timing of reading and writing transactions on the blockchain. Teechain maintains fund deposits for off-chain payment channels by using secure and trusted treasuries, which are protected by implementing them inside TEEs. By trusting the TEEs, treasuries can adopt a new efficient off-chain payment protocol that simplifies both payment and finalizing payment. To make Teechain robust against TEE failures or compromises, the state of each treasury is replicated among a small committee. In each committee of treasuries, a treasury must obtain approvals from a subset of other committee treasuries to be able to issue an off-chain transaction or finalize a payment channel. Hence, the efficiency of payment channels as a whole is improved by the TEEs, but the security guarantees of Teechain do not depend on each individual TEE.

5 Conclusions

In this chapter, we have surveyed existing literature on recent advances in the security of blockchain. In particular, we have shown several recent attacks, especially on network and smart contacts, and reviewed some security enhancement solutions for blockchain. It is suggested that the blockchain technology is still susceptible to various attacks that could obstruct an entire system and potentially cost hundreds of millions of dollars. Therefore, despite the great potential of blockchain, it is still in its early stage and a lot of research effort is needed to realize the vision of a decentralized platform for various applications.

References

1. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2009). http://www.bitcoin.org/bitcoin.pdf
2. Conti, M., Kumar, E.S., Lal, C., Ruj, S.: A survey on security and privacy issues of bitcoin. IEEE Commun. Surv. Tutor. **20**(4), 3416–3452 (2018)
3. Castro, M., Liskov, B., et al.: Practical byzantine fault tolerance. OSDI **99**(1999), 173–186 (1999)
4. Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: a distributed operating system for permissioned blockchains. In: Proceedings of the Thirteenth EuroSys Conference, pp. 1–15 (2018)

5. Nguyen, T.D., Thai, M.T.: A blockchain-based iterative double auction protocol using multi-party state channels. ACM Trans. Internet Technol. (TOIT) **21**(2), 1–22 (2021)
6. Christidis, K., Devetsikiotis, M.: Blockchains and smart contracts for the internet of things. IEEE Access **4**, 2292–2303 (2016)
7. Zhou, S., Möser, M., Yang, Z., Adida, B., Holz, T., Xiang, J., Goldfeder, S., Cao, Y., Plattner, M., Qin, X., et al.: An ever-evolving game: evaluation of real-world attacks and defenses in ethereum ecosystem. In: 29th {USENIX} Security Symposium ({USENIX} Security 20), pp. 2793–2810 (2020)
8. Kalra, S., Goel, S., Dhawan, M., Sharma, S.: Zeus: analyzing safety of smart contracts. In: NDSS (2018)
9. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: IEEE Symposium on Security and Privacy (SP), vol. 2016. IEEE, 839–858 (2016)
10. Zhang, M., Zhang, X., Zhang, Y., Lin, Z.: {TXSPECTOR}: uncovering attacks in ethereum from transactions. In: 29th {USENIX} Security Symposium ({USENIX} Security 20), pp. 2775–2792 (2020)
11. Merkle, R.: One way hash functions and des. In: Conference on the Theory and Application of Cryptology, pp. 428–446. Springer (1989)
12. Diffie, W.: The first ten years of public-key cryptography. In: Proceedings of the IEEE, pp. 560–577 (1988)
13. Garfinkel, S.: Public key cryptography. Computer **29**(6), 101–104 (1996)
14. Schollmeier, R.: A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In: Proceedings First International Conference on Peer-to-Peer Computing, pp. 101–102. IEEE (2001)
15. Badreddin, O., Gomez Rivera, A., Malik, A.: Blockchain fundamentals and development platforms. In: Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, pp. 377–379. ACM (2018)
16. Penard, W., van Werkhoven, T.: On the secure hash algorithm family. In: Cryptography in Context, pp. 1–18 (2008)
17. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: 24th {USENIX} Security Symposium ({USENIX} Security 15), pp. 129–144 (2015)
18. Eyal, I., Sirer, E.G.: Majority is not enough: Bitcoin mining is vulnerable. In: International Conference on Financial Cryptography and Data Security, pp. 436–454. Springer (2014)
19. Karame, G.O., Androulaki, E., Capkun, S.: Double-spending fast payments in bitcoin. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security, pp. 906–917 (2012)
20. Apostolaki, M., Zohar, A., Vanbever, L.: Hijacking bitcoin: routing attacks on cryptocurrencies. In: IEEE Symposium on Security and Privacy (SP), vol. 2017, pp. 375–392. IEEE (2017)
21. Saad, M., Cook, V., Nguyen, L., Thai, M.T., Mohaisen, A.: Partitioning attacks on bitcoin: colliding space, time, and logic. In: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 1175–1187. IEEE (2019)
22. Tran, M., Choi, I., Moon, G.J., Vu, A.V., Kang, M.S.: A stealthier partitioning attack against bitcoin peer-to-peer network. In: IEEE Symposium on Security and Privacy (S&P) (2020)
23. Apostolaki, M., Marti, G., Müller, J., Vanbever, L.: Sabre: protecting bitcoin against routing attacks. In: NDSS Symposium (2019)
24. Permenev, A.,Dimitrov, D., Tsankov, P., Drachsler-Cohen, D., Vechev, M.: Verx: safety verification of smart contracts. In: 2020 IEEE Symposium on Security and Privacy, SP, pp. 18–20 (2020)
25. Torres, C.F., Steichen, M., et al.: The art of the scam: Demystifying honeypots in ethereum smart contracts. In: 28th {USENIX} Security Symposium ({USENIX} Security 19), pp. 1591–1607 (2019)
26. Perez, D., Livshits, B.: Broken metre: attacking resource metering in evm. In: NDSS Symposium (2019)

27. Steffen, S., Bichsel, B., Gersbach, M., Melchior, N., Tsankov, P., Vechev, M.: zkay: specifying and enforcing data privacy in smart contracts. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, pp. 1759–1776 (2019)
28. Mirkin, M., Ji, Y., Pang, J., Klages-Mundt, A., Eyal, I., Jules, A.: Bdos: blockchain denial of service. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (2020)
29. Bonneau, J.: Hostile blockchain takeovers (short paper). In: International Conference on Financial Cryptography and Data Security, pp. 92–100. Springer (2018)
30. Bag, S., Ruj, S., Sakurai, K.: Bitcoin block withholding attack: analysis and mitigation. IEEE Trans. Inf. Forens. Secur. **12**(8), 1967–1978 (2016)
31. Kwon, Y., Kim, D., Son, Y., Vasserman, E., Kim, Y.: Be selfish and avoid dilemmas: fork after withholding (faw) attacks on bitcoin. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 195–209 (2017)
32. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: International Conference on Financial Cryptography and Data Security, pp. 6–24. Springer (2013)
33. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: International Conference on Financial Cryptography and Data Security, pp. 34–51. Springer (2013)
34. Fanti, G., Viswanath, P.: Deanonymization in the bitcoin p2p network. In: Advances in Neural Information Processing Systems, pp. 1364–1373 (2017)
35. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: decentralized anonymous payments from bitcoin. In: IEEE Symposium on Security and Privacy, vol. 2014, pp. 459–474. IEEE (2014)
36. Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct non-interactive arguments via linear interactive proofs. In: Theory of Cryptography Conference, pp. 315–333. Springer (2013)
37. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles, pp. 51–68 (2017)
38. Eyal, I., Gencer, A.E., Sirer, E.G., Van Renesse, R.: Bitcoin-ng: a scalable blockchain protocol. In: 13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16), pp. 45–59 (2016)
39. Miller, A., Xia, Y., Croman, K., Shi, E., Song, D.: The honey badger of bft protocols. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 31–42 (2016)
40. Nguyen, T., Thai, M.T.: Denial-of-service vulnerability of hash-based transaction sharding: attack and countermeasure. In: (2020). arXiv preprint arXiv:2007.08600
41. Anati, I., Gueron, S., Johnson, S., Scarlata, V.: Innovative technology for CPU based attestation and sealing **13**, 7 (2013)
42. Cheng, R., Zhang, F., Kos, J., He, W., Hynes, N., Johnson, N., Juels, A., Miller, A., Song, D.: Ekiden: a platform for confidentiality-preserving, trustworthy, and performant smart contracts. In: IEEE European Symposium on Security and Privacy (EuroS&P), vol. 2019, pp. 185–200. IEEE (2019)
43. Matetic, S., Wüst, K., Schneider, M., Kostiainen, K., Karame, G., Capkun, S.: {BITE}: Bitcoin lightweight client privacy using trusted execution. In: 28th {USENIX} Security Symposium ({USENIX} Security 19), pp. 783–800 (2019)
44. Lind, J., Naor, O., Eyal, I., F. Kelbert, Sirer, E.G., Pietzuch, P.: Teechain: a secure payment network with asynchronous blockchain access. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles, pp. 63–79 (2019)
45. Dinh, T.T.A., Liu, R., Zhang, M., Chen, G., Ooi, B.C., Wang, J.: Untangling blockchain: a data processing view of blockchain systems. IEEE Trans. Knowl. Data Eng. **30**(7), 1366–1385 (2018)
46. Rogaway, P., Shrimpton, T.: Cryptographic hash-function basics: definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: International Workshop on Fast Software Encryption, pp. 371–388. Springer (2004)

47. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Blockchain Technology Overview. Cornell University (2019). arXiv preprint arXiv:1906.11078
48. Hafid, A., Hafid, A.S., Samih, M.: Scaling blockchains: a comprehensive survey. In: IEEE Access, vol. 8, pp. 125 244–125 262 (2020)
49. Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. IEEE Commun. Surv. Tutorials **18**(3), 2084–2123 (2016)
50. Kogias, E.K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing bitcoin security and performance with strong consistency via collective signing. In: 25th {usenix} Security Symposium ({usenix} Security 16), pp. 279–296 (2016)
51. Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30 (2016)
52. Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B.: Omniledger: A secure, scale-out, decentralized ledger via sharding. In: IEEE Symposium on Security and Privacy (SP), vol. 2018. IEEE, 583–598 (2018)
53. Zamani, M., Movahedi, M., Raykova, M.: Rapidchain: scaling blockchain via full sharding. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 931–948 (2018)
54. Dziembowski, S., Faust, S., Hostáková, K.: General state channel networks. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 949–966 (2018)
55. Miller, A., Bentov, I., Bakshi, S., Kumaresan, R., McCorry, P.: Sprites and state channels: payment networks that go faster than lightning. In: International Conference on Financial Cryptography and Data Security, pp. 508–526. Springer (2019)
56. Malavolta, G., Moreno-Sanchez, P., Kate, A., Maffei, M., Ravi, S.: Concurrency and privacy with payment-channel networks. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 455–471 (2017)
57. Dziembowski, S., Eckey, L., Faust, S., Malinowski, D.: Perun: virtual payment hubs over cryptocurrencies. In: IEEE Symposium on Security and Privacy (SP), pp. 106–123. IEEE (2019)
58. Malavolta, G., Moreno-Sanchez, P., Schneidewind, C., Kate, A., Maffei, M.: Anonymous multi-hop locks for blockchain scalability and interoperability. In: NDSS (2019)
59. Mavroudis, V., Wüst, K., Dhar, A., Kostiainen, K., Capkun, S.: Snappy: fast on-chain payments with practical collaterals. In: NDSS (2020)
60. Li, P., Miyazaki, T., Zhou, W.: Secure balance planning of off-blockchain payment channel networks. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 1728–1737. IEEE (2020)
61. Khalil, R., Gervais, A.: Revive: rebalancing off-blockchain payment networks. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 439–453 (2017)
62. Yu, H., Nikolić, I., Hou, R., Saxena, P.: Ohie: blockchain scaling made simple. In: IEEE Symposium on Security and Privacy (SP), pp. 90–105. IEEE (2020)