Received: 4 September 2018 | Revised: 25 October 2018 | Accepted: 13 November 2018

DOI: 10.1002/nem.2060

RESEARCH ARTICLE

WILEY

A survey on blockchain cybersecurity vulnerabilities and possible countermeasures

Huru Hasanova | Ui-jun Baek | Mu-gon Shin | Kyunghee Cho | Myung-Sup Kim

Department of Computer and Information Science, Korea University, Sejong, Republic of Korea

Correspondence
Myung-Sup Kim, Department of Computer and Information Science, Korea University, Sejong, Republic of Korea.
Email: tmskim@korea.ac.kr

Funding information
Institute for Information & Communications Technology Promotion (IITP), Grant/Award Number: 2018-0-00539-001

Summary

Blockchain technology has attracted considerable attention owing to its wide range of potential applications. It first appeared as a cryptocurrency, called Bitcoin, but has since been used in many other business and nonbusiness applications. Unlike most existing systems that are based on centralized frameworks, this new technology utilizes peer-to-peer networks and distributed systems which includes blockchain registers to store transactions. Its structure is designed as a digital log file and stored as a series of linked groups, called blocks. Each individual block is locked cryptographically with the previous block. Once a block has been added, it cannot be altered. Many security experts speculate that the inherent cryptographic nature of the blockchain system is sufficient to withstand constant hacking and security threats. However, previous studies on the security and privacy of blockchain technology have shown that many applications have fallen victim to successful cyberattacks. Owing to the increasing demand for cryptocurrency and its current security challenges, previous studies have not focused on blockchain technology cybersecurity vulnerabilities extensively. Here, our study extends upon the previous studies on vulnerabilities and investigates the types of potential attacks. Our study then provides further direction to highlight possible countermeasures against blockchain technology vulnerability to cybersecurity.

1 | INTRODUCTION

Blockchain technology (BT) is a decentralized transaction and data management technology that provide security, anonymity, and data integrity without involving any third-party organization in control of the transactions.[1] BT has equity management capabilities by using electronic invoice ledgers for transactions performed over the internet.[2] BT is also being applied in the fields of finance, gaming, gambling, supply chain, manufacturing, trade, and e-commerce.[3] BT system is an immutable database of all historical transactions stored as a digital ledger. Furthermore, all nodes (users) on the distributed blockchain network can manage the shared ledger.[4] Blocks are arranged in chains, where the bottom block is the foundation of the stack. Each block is linked to the preceding block in the chain.[5] Using cryptographic hash algorithms, each block is identified by a generated hash.[6] A block in the chain can have one parent block, but multiple child blocks.[7] A block contains a header, made up of a unique hash of its parent blocks that connects it with its parent blocks, forming a chain. The first block is known as a genesis block (the first Bitcoin block was created in 2009).[8] Thus, BT system is a digital record of ownership that works as a decentralized database system, to which a continuously growing list of transaction records is maintained, which differs from traditional centralized database systems.

 © 2019 John Wiley & Sons, Ltd.

The use of the BT in Bitcoin, which was launched in November 2008,[9] is largely responsible for the growing interest in BT. Bitcoin, a decentralized peer-to-peer digital currency, tracks all digital events in a public ledger. It records all transactions that are shared between participating parties and are verified by a consensus of participants in the shared system. Once information has been recorded during a digital event, it cannot be altered. Thus, Bitcoin contains a continuous and verifiable record of every event. In terms of security, Bitcoin is highly controversial in the digital currency market. However, BT has found a wide range of applications in both financial and nonfinancial sectors. A blockchain creates a distributed consensus in the digital world, providing entities with a secure platform that maintains past records of digital events by creating an irrefutable record in a public ledger.[10]

Activities associated with BT are classified into three categories in the viewpoint of organization and accessibility: (a) the first-generation public blockchain (blockchain 1.0), (b) the second-generation public blockchain (blockchain 2.0), and (c) the third-generation private blockchain (blockchain 3.0).[11] Blockchain 1.0 deploys cryptocurrencies in applications related to cash, such as currency transfers, currency settlements, and digital payments. Blockchain 2.0 includes smart contracts for economic markets and financial applications. This category handles more than simple cash transactions. It includes stocks, bonds, loans, mortgages, titles, smart properties, and smart contracts. The third category applies to applications beyond currencies, finance, and markets. It includes areas, such as government, health, science, literacy, culture, and art. Therefore, blockchains within this category are considered private.[12] Blockchain is a promising technology that may alleviate the risk of cyberattack directed to a single point, which could bring down the entire network.[13] However, a coded intrusion or system vulnerability could allow more negative consequences to the security of the system. For example, if successful, an attacker would gain access not only to the information stored at the point of attack but also to all information recorded in the ledger. Thus, security issues related to blockchain are critical in terms of cybersecurity. In this sense, security experts need to fully understand the scope and impact of the security and privacy challenges related to blockchain before predicting the potential damage from an attack, and verify whether the current technology can withstand persistent hacking attempts.

Previous studies have explored the technical architecture of BT in relation to cryptocurrency.[14] Although some studies have focused on the security aspects of BT, owing to the increasing demand for cryptocurrency with its current security challenges, these studies have focused little on BT cybersecurity vulnerabilities. Among these studies, Conti et al[15] focused on the fundamental background of Bitcoin cryptocurrency, overviews of its use and functionalities, and its privacy aspects. Atzei et al[16] analyzed Ethereum smart contracts and offered taxonomy of general programming pitfalls and bugs related to BT vulnerabilities. With the emergence of dynamic development attack methods by the hackers, the existing approaches to security are becoming outdated and less effective. Based on such grounds, our study presents a comprehensive review of BT security vulnerabilities by exploring attack vectors that focus on user security and its vulnerabilities. The three main contributions of our study are as follows: First, our study examines the security challenges and problems of existing cryptocurrencies, including the possibility of attacks, focusing primarily on issues of user privacy and transaction anonymity. Note that our study does not attempt to solve these challenges and threats, but instead presents an overview of blockchain security, including examining its vulnerabilities and discussing possible countermeasures. We investigate, at various levels, the types of attacks that pose both practical and theoretical risks to BT. Second, according to our study, we discuss the limitations of the state-of-the-art solutions that address security threats and enable strong privacy. Third, based on our thorough review, we then provide possible directions for further research on countermeasures for BT security vulnerabilities.

This paper is organized as follows. Section 2 presents a comprehensive overview of security risks and explores real attack cases of proof of work (PoW) and proof of stake (PoS) based BT. Section 3 discusses a number of security threats associated with the development, implementation, and use of Smart contract–based BT. Section 4 demonstrates the security issues and possible security threats of private BT. Section 5 discusses the proposed state-of-the-art solutions that counter various security threats and enhance existing security issues related to private and public BT. Section 6 concludes with suggestions and future directions for possible countermeasures against BT cybersecurity vulnerabilities. Section 7 concludes our study.

2 | VULNERABILITIES OF BT

BT systems are based on an append-only data structure that stores every transaction executed over a network. Each block in the network includes a link (hash) to its predecessor, creating a string of blocks that maintains a link to the genesis block. Owing to the parent-child relation between the blocks, all subsequent blocks are mined when a new

block is added to the network chain. It is methodologically complex to reverse a block retrospectively once it has been on the chain for some time. Nodes in a PoW-consensus architecture are decentralized and communicate over a network, working in concert to construct a blockchain. Interestingly, this offers a state similar to an anonymous network, because it does not require permission from a trusted third party to process a transaction. The distributed maintenance of the blockchain creates a system with complete transparency. In this case, all processed transactions remain transparent within the system and are validated before they are added to the blockchain. This can significantly reduce users' ability to "double spend" on their respective digital assets. However, it is still possible for an attacker with high hashing power to insert invalid transactions into a block. In this case, the attacker would gain full access to the network, allowing him/her to deny service to specific participants. Network security experts call this a 51% attack, which can leave a blockchain extremely vulnerable. Therefore, a PoW consensus architecture is vulnerable to 51% attacks when a mining pool can control 51% of the hash rate.

With the emergence of Bitcoin, PoW became the most common architecture used for authenticating a cryptocurrency structure (peer-to-peer system). However, it has limitations, particularly concerning the difficulty and complexity of mining and high level of energy consumption. As an alternative to PoW, proof-of-stake (PoS) architecture has been proposed, which relies on a certifier's economic investment in the network.[17] PoS replace the mining operation with an alternative approach based on a user's stake or ownership of a virtual currency in the BT system. In a PoW architecture, a user invests in mining equipment and earns a mining reward for validating transactions. In contrast, the PoS architecture allows users to buy cryptocurrency as a stake in the blockchain system proportional to their investment, allowing them to participate in block formation as a certifier. The PoS architecture selects certifiers in a random manner when creating blocks. Thus, no certifier can predict its turn in advance.[18] However, PoS have one major drawback, called nothing-at-stake.[19] In the event of a fork, whether it was unplanned or as a result of a malicious attempt to rewrite the history of the chain and reverse a transaction, the optimal strategy for any miner is to mine on every chain, thus ensuring that he/she receives a reward, regardless of which fork wins. In case a large number of economically interested miners are assumed, an attacker may be able to send a transaction in exchange for some digital good and receive the good. The fork will initiate of the blockchain transaction from one block behind and send the money instead. Even with 1% of the total stake, the attacker's fork would win because every node is mining on both. With PoS method, Ethereum tried to overcome PoW security limitation, a reduced risk of centralization, and energy efficiency.[20] So that, PoS allows a consensus to be established on the network, whereas the PoW architecture has many potential security issues.[21] As a more comprehensive view, we present these potential security issues, along with their impacts on PoW and PoS blockchain architectures in Table 1 in Appendix A section.

The following sections explain the current vulnerabilities that threaten the widespread use of cryptocurrencies and BT, focusing particularly on those found in the consensus architectures of PoW, PoS, DPoS, and Smart contracts. We systemized vulnerabilities by blockchain type, consensus type, and attack types (see Figure 1).

2.1 | General risks of blockchain 1.0 and 2.0

2.1.1 | Double spending

Double spending happens when a user makes multiple payments using one particular funding form, possibly in a P2P network. This occurs because transactions are validated by solving a mathematical problem. Hence, a time lapse is required before transactions are confirmed. When unprocessed payments are announced over the network, or when the network's nodes are updated with transactions that are yet to be confirmed, broadcasting disruptions are possible at different time slots. In Bitcoin, it typically takes approximately 10 minutes to solve a problem successfully. This is dictated by the difficulty of the calculation, which involves adjustments to the processing power of the miners.[22] Karame et al[23] explain this as follows. Suppose an attacker A needs to trick a retailer R into accepting a transaction Tr_R that R will not be able reverse. In this case, A creates another transaction Tr_A with the same inputs as Tr_R, but replaces the recipient address under the control of A. If both transactions are initiated at the same time to peers on the chain, then the chain will not accept multiple transactions that share common inputs. They will only accept the version of the transaction that reaches them first for inclusion in the generated blocks, ignoring all other versions. This means a double-spending attack can be performed successfully if R receives Tr_R, but the majority of the peers in the network receive Tr_A, which then has a significant possibility of being included in a generated block. In order to perform a successful double-spending attack, a few basic conditions and requirements are necessary. First, the time $t_R^R < t_R^A$ otherwise R will

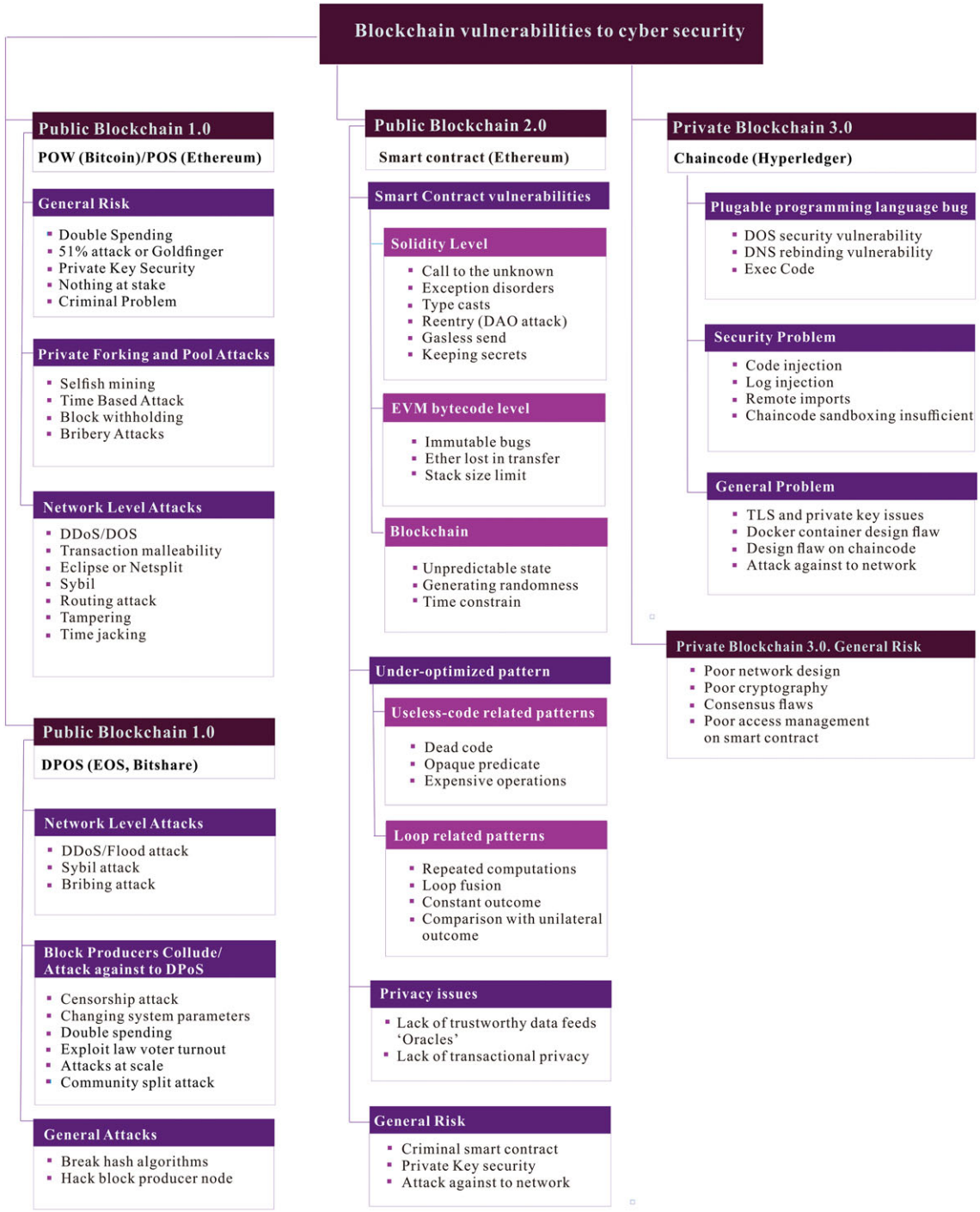

FIGURE 1 Vulnerabilities in BT

first add Tr_A in a memory pool and will reject other inputted transactions with Tr_A. In all cases, Tr_A will be confirmed as the first block to be added to the chain and other attacks will fail.[23] The above explanation and the principle of a double-spending attack are shown in Figure 2.

2.1.2 | The 51% attack or Goldfinger

Blockchain consensus architectures are particularly vulnerable to double spending and 51% attacks. These attacks cannot be avoided in such systems and, theoretically, can happen all the time. The 51% attack (or Goldfinger) was first used to attack Bitcoin, but can also be used on other BT systems. When good nodes control at least 51% of the network

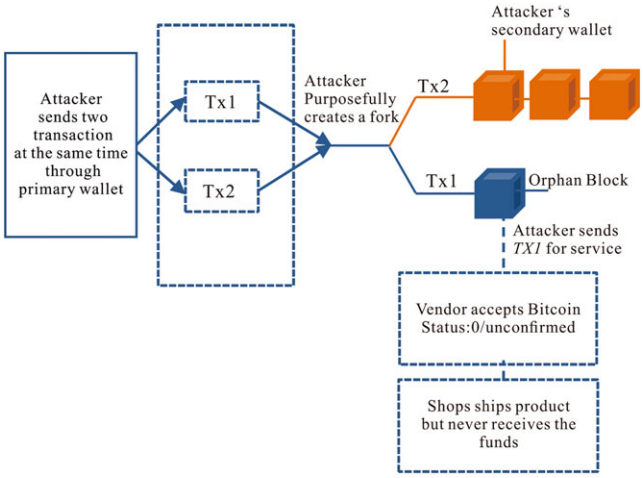

FIGURE 2 Working principles of double-spending attack

mining power, a BT system can then be considered protected. In this case, the cost of controlling a major stake might be greater than the cost of obtaining significant mining power, thus increasing the cost of attacks. In addition, the attacker's coin time can be paid during the attack, which may make it more difficult for the attacker to prevent transactions from being added to the leading chain.[24] An attacker might also be able to obstruct the confirmation of new transactions on the network, thus preventing some or all nodes from receiving funds. In addition, an attacker can reverse transactions, but only if they have dominant power over the network, resulting in double spending. In the event of a 51% attack, it would be extremely difficult for an attacker to take over the blockchain, because transactions are locked prior to the start of an attack if an attempt is made to change historical blocks.

In July 2014, the mining pool ghash.io briefly exceeded 50% of the Bitcoin network's processing power, which led to the pool reducing its share of the network.[25] In August 2016, two BT-based systems Ethereum Krypton and Shift suffered 51% attacks. An attempt was made to overwhelm the network with at least 51% of the hashing power in order to roll back the transactions and spend the same coins again. BT-based Ethereum has faced 51% attacks, but mainly in low hashing power attacks. A severe case occurs when the attacker has more than 67% of the stake by which the attacker can freely block any transactions and wish to block and reject to form any blocks of the transactions.[19]

2.1.3 | Wallet security (private key security)

Generally, cryptocurrencies store their value in a file store called a wallet, whereby each client owns a set of private-public keys to access the wallet. The major weakness with the wallet is that it can be influenced, pinched, and relocated just like other stores. Users often fail to recall their protective PIN or password, or lose the hard drive where the private key is located. This means that a user may not always be able to access their store. In light of this, ransomware can cause the same issue. Wallet theft uses classic mechanisms such as phishing, which include system hacking, the installation of buggy software, and the incorrect use of wallets.[26]

A blockchain system can easily be exploited through any vulnerability that might contribute to a cryptographic solution, because it is obvious that any programming bug or lack of secure private key can be the foundation of a major security breach. Hypothetically, a cryptoattacker should not be able to understand the original plain text, which is encrypted. However, it is not difficult to understand the format of the blocks, and even a good cryptograph makes a plain text, such as random gibberish, but certain characters or numbers are often found in the same place in each block in the blockchain. This allows an attacker the opportunity to attempt a partial representation of the plain text in every cryptoprotected block, where each block is a function of the preceding block.[27]

In the cryptocurrency domain, Bitcoin has the largest market share, where a Bitcoin wallet employs a public key, private key, and an individual address. According to VanDam and Shparlinski, public keys can be generated safely from private keys using an algorithm called elliptic curve digital signature (ECDSA).[28] However, Vedral and Morikoshi argue that quantum computers can crack ECDSA.[29] In addition, a machine can exploit quantum weirdness because its underlying reality is still unknown. This could allow the presence of quantum bits (qubit), as well as algorithms that quantum computers can perform that classical computers cannot.[29] For example, a quantum computer can run Shor's algorithm

and quickly crack any public key encryption by finding the factors of large numbers.[30] Ironically, the Bitcoin protocol address is derived using the SHA-256 function for public keys, using the RIPEMD-160 hash function and adding a checksum for error correction. While the mathematical weaknesses of SHA-256 are surprising,[31] no SHA-256 cracking incidents have occurred, and thus, it has a strong and predictable future.

2.2 | Specific flaw in PoS

When a stake moves, the existing majority of stakeholders remain honest in that past account keys that have no current stake can be negotiated. This is a major weakness in PoS-based blockchains because a set of malicious shareholders can create a different blockchain using old accounts, creating a blockchain is relatively straightforward.[32] In general, a nothing-at-stake attack against a PoS-based blockchain is simplified by shareholders maintaining several blockchains concurrently, manipulating the fact that very little computational power is required to create a PoS-based blockchain. Stakeholders have an incentive to act properly in a stake on the longest chain in order to preserve the value of their investment. However, this ignores several problems. A stakeholder has only a 1% probability of being a critical part of an attack, which will otherwise fail. In light of this, a bribe is required to induce individual stakeholders to join an attack that would be only 1% of the size of their deposit. Hence, the required shared bribe can only be 0.5% to 1% of the total deposit. This assumption implies that a zero-chance-of-failure situation cannot be a constant equilibrium, because if the probability of failure is zero, then everyone has a 0% probability of being critical. The PoS architecture is also vulnerable to a long-range attack, in which an attacker with 1% of all coins initiates a fork without the most recent blocks on the main chain. The attacker endeavors to start a fork after the genesis block, easily creating new blocks and simply making the longest chain. In such cases, a new user cannot identify the longest blockchain, because many blocks are illegal. One possible solution to this is to have every block contain a timestamp, where a user can reject chains with timestamps that are too far ahead of its own timestamp. This constitutes a short-range attack because it occurs rarely and would later disappear. In such cases, the attacker can hide the generated blocks until they can be considered a local maximum, and then reveal the best chain and execute it more frequently. Thus, the attack rises locally and is then repeated. Even with a 30% stake, the attacker can create 28 to 30 portions of blocks in a 500-step run. However, within a 1000-step interval, the chance of an attack is wiped out. Subsequently, the local attack with the best blockchain ends up with an order of blocks created by the attacker can easily be detected. As presented above, a long-range attack is questionable owing to the retargeting procedure and chain measurement. However, the possibility of a short-range attack from a multibranch account still exists, although it is impossible across the whole multibranch environment.[19]

2.3 | Private forks and pool attacks

The PoW-based blockchain is widely used in the digital currency system, where it has taken over a large part of the total market capitalization. The security of a PoW architecture is absolutely dependent on the state of the security of the blockchain system, in which attacker side, if any, should have 50% or less of the computational control. Adding to this, Eyal and Sirer offered a selfish mining, or blocked discarding attack, that proved that conventional wisdom is incongruous, and that certain attacks enable a selfish miner equipped with 25% to up to 33% (hard theoretical boundary) of the extracting power that can allow an attacker to earn 50% of mining power.[33,34] Moreover, if the attacker is supervised at a high level, it will not perform as a normal miner. Rather, they will immediately be able to publish blocks on the network or assess the system to publish the blocks in a selective manner. Meanwhile, the attacker may choose to forgo his/her own revenue. However, if too many blocks are published at once, this may lead to a rejection of blocks that will culminate in a reduction of revenue. In this case, the reduction in revenue is temporary and short-lived for the attacker, while it negatively affects the revenue of others. Subsequently, incentives are created for impartial nodes to consolidate their efforts with that of the attacker, forming an alliance that will increase their revenue. This alliance would increase the population of the attacker's alliance by 50%, offering the attacker a higher level of control over the network. As such, the attacker's strategy would be to create a private chain that could be uniquely distinguished from a public string. At first, this private and public chain are initiated together, while the attacker consistently seeks opportunities to mine on the private chain and secure retention rights on the private blocks available. This tactic helps the attacker to decide on the perfect time to publish a block. This can be explained further using the scenario presented below.

Let us assume that X is an attacker's portion of the network hash power. Two existing public chains compete for control of a portion of the network that affects the revenue of the attacker chain (Z). Then, we have the following situation:

- State 0: Represents a situation where the attacker's private chain is related to the public chain; hence, the mining should be performed on the private chain. When the private chain block 1 is ahead, there is a probability of X that the attacker will encounter a block and move to state 1, while there is a probability of 1 − X that the public network will have to find a block and the attacker will have to reorganize his or her private network as a public chain.
- State 1: On the other hand, this represents a case where the attacker's private chain is ahead of the public chain by 1. Here, the mine should be carried out on a private chain. When the private chain block 2 is ahead, there is a probability of X that the attacker will move to state 2, while there is a probability of 1 − X that the public network will find a block and set it to a state of $0^{'}$.
- State $0^{'}$: In this situation, the blocks of the attacker must be published, which may result in two competing chains, each of which are one block longer. Given a probability of X, the attacker is required to find another block that can force the network to become a private network. There is a possibility that the attacker will obtain a revenue of 2 and that the system will be forced to reset to state 0. In this situation, both the attacker's network and the public network have a chance of generating a revenue of 1 when the system is reset to state 0. There is a probability of $(1 - X)(1 - Z)$ that the network will find another network with blocks other than its own to gain a revenue of 2. If this happens, the system is reset to a state of 0.
- State 2: The attacker may progress to a state of 3 and may earn a revenue of 1 with a probability of X. Even though the revenue gains will be derived later, accountability is easier in this case. There is a probability of 1 − X that it is possible for the network to find a block while the attacker will have to publish 2 blocks in his or her private chain, which may be longer than the public chain by 1. Finally, to obtain a revenue of 2, the network can choose to switch to the attacker's chain.
- State i ($i > 2$): The attacker may move to a position of $i + 1$ and gain a revenue of 1 at a probability of X, while there is a probability of 1 − X that the attacker will return to a state of $i - 1$.

Next, we examine a confirmatory test for the scenarios listed above. Suppose Z is a number close to one, in which case the chance of an attacker discarding a block is slim. This is only possible in the time frame of state $0'$ and if $Z \approx 1$ across the entire network. This is because at $Z < 1$, other nodes in addition to the attacker can mine on the attacker's block. Hence, discarding a block is unreasonable, and so, the attacker has to mine optimally. However, blocks that have been cast off can be found by the public network in states $0'$ and 2 if the public network is mined with partial efficiency. As a result, nonaligned (profit-maximizing) nodes are enticed to form a coalition with the attacker's team to increase their gains. While Z decreases, this benefits the attacker by $Z = 0.5$. Eyal and Sirer have shown successfully that the attacker is more competent than the public network at $X > 1/4$ and $X > 1/3$. Furthermore, the public network is less effective than the attacker at any Z. Next, we consider how to calculate Z. Presently, the Bitcoin network has been enhanced by tracking the activities of nodes that merely mine and propagate the first block they encounter. This has the potential to prevent attacks because the approach of the attacker is reactive, and Z decreases to zero as individual blocks are discarded from a public network. Nevertheless, a "Sybil attack" can be carried out by well-financed attackers (or attackers controlling botnets) by creating millions of nodes and inserting them into the network in many places. When an attack takes place, these nodes (Sybil) concentrate on propagating the attacker's blocks alone, which, as Eyal and Sirer show, make Z tend toward one. In reality, however, this is not true, because $Z \leq 0.8$ is ensured when a minimum individual mining pool can catch their own blocks first. There are mixed views on whether Sybil attacks can be carried out in this state. To guarantee the security of Bitcoin against Sybil attacks, Eyal and Sirer argue that strategic switching be used by honest miners, which involves the propagation of all blocks when multiple competing chains of the same length are received. However, if this is implemented by all miners, then $Z = 0.5$, and a reasonable threshold of $X \geq 1/4$ for the attack is possible.[33] Figure 3 illustrates the process of a selfish attack as a state machine in a systematic order.

A "bribery attack" is a form of mining attack. Bonneau et al[34] discussed three ways in which such attacks could occur. First, in an out-of-band payment, owners of mining capacity are paid openly. The attacker is required to pay slightly above market price to ensure having more than half the total hash power, which makes the attack statistically effective. The second is a negative-fee mining pool, and the third is an in-band payment via forking, in which the attacker endeavors to bribe using Bitcoin by creating a fork that covers the bribe money accessible to any miner embracing the fork. The cost of this attack can be randomly minor, but it requires a lot of a capital. However, it is unsafe and

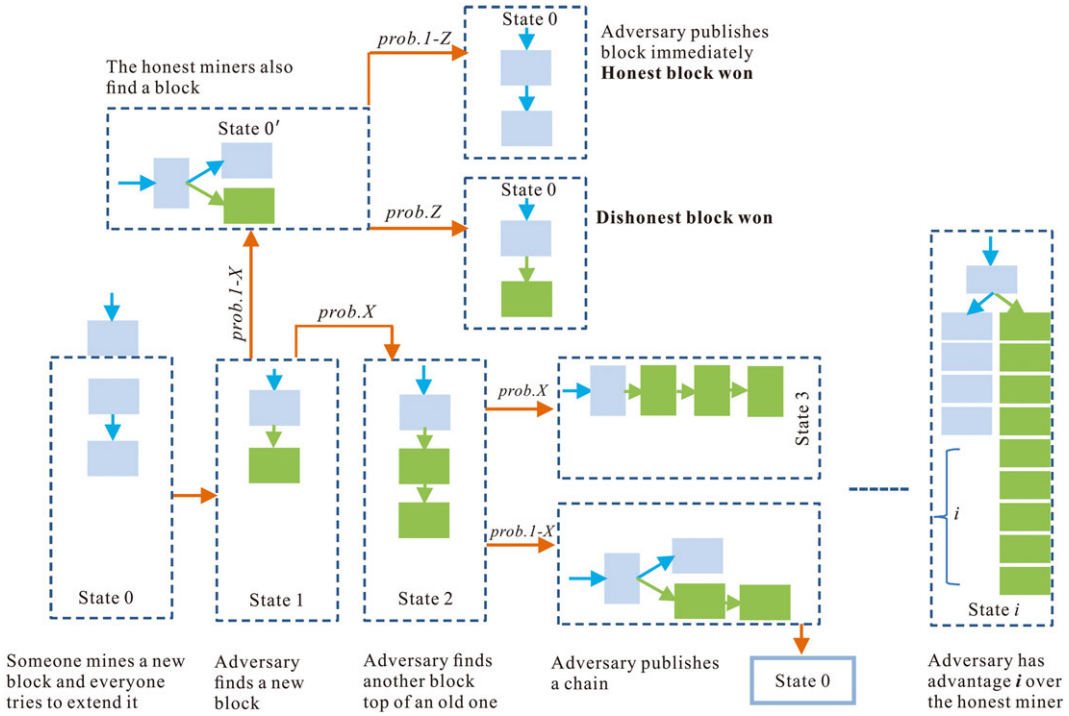

FIGURE 3 State machine a selfish attack

appears to be difficult because miners do not merely rent their hash power to the highest bidder when they can use mining pools that they trust. Thus, there is probably no way for an attacker to obtain more than the 50% of the total hash power required to make the attack effective.[34]

2.4 | Network-level attack

At present, blockchain network security problems become the most popular research issues on the network security field. However, there are still various concerns about its scalability, security, availability, and sustainability. With the rise of the digital currency market, cyber attacks that endeavor to influence marketing and business-oriented services are constantly increasing. Among the numerous attacks, distributed denial of service (DDoS) attacks are one of the most common network bandwidth consumption attacks that have caused trouble for services. DDoS attacks on blockchain-based platforms are not like regular attacks. In a decentralized and peer-to-peer technology, it is more difficult and costly than in conventionally distributed application architecture when an endeavor to subdue the network using a large volume of small transactions occurs. However, blockchain-based platforms, such as Ethereum and Bitcoin, are vulnerable to DDoS attack. Therefore, there are adequate protection measures needed both at the network and application level. In a case example, in 2016, the Ethereum and Bitcoin networks endured the same DDoS attacks as in 2014. Thus, resilient and decentralized blockchain solutions can offer high accessibility, but DDoS attacks will remain a determined danger to security.[16]

In a cryptocurrency ecosystem, currency exchanges play a paramount role, but in most cases, these systems encounter DDoS attack more frequently. Feder et al[35] stated that several currency exchanges have been shut down as a result of DDoS attacks. Mt. Gox is one of the main exchanges that handles more than 70% of all Bitcoin transactions worldwide. It is a leading Bitcoin intermediary, and is considered the biggest Bitcoin exchange. Vasek and Moore carried out a wide empirical analysis of DDoS attacks in the Bitcoin ecosystem, and reported 58 attacks on exchanges and Bitcoin services. In particular, there have been 142 unique DDoS attacks on 40 Bitcoin services, where 7% of all known operators have faced attacks. The authors also report that exchanges, mining pools, gambling operators, wallets, and financial services are far more vulnerable to DDoS attacks than other services are. Other reports indicate that 17.1% of small mining pools have been affected by DDoS attacks, whereas 62.5% of large pools have faced similar attacks.[36] Johnson et al[37] studied a series of game-theoretical architectures of competition between two pools of varying sizes in order to carry out a DDoS attack against another mining pool. They compared an honest approach with a dishonest strategy. Under the honest

paradigm, players of a coalition could invest in additional computing resources to increase the likelihood of winning the next race. Dishonest player coalitions focused on a mining pool and triggered a costly DDoS attack to lower the expected success of a competing mining pool. The authors made two other observations as to whether there was a greater incentive to attack a larger mining pool than a smaller one. Each pool battle was designed for reward. Thus, eliminating the largest mining pool had the greatest effect on the chance of the remaining mining pools winning. Second, a larger mining pool had a relatively smaller competitor base, and eliminating a competitor from a small base yielded a greater benefit than eliminating a competitor from a larger base. Therefore, there is a size threshold beyond which mining pools are more likely to be subject to economically motivated attacks. Moreover, players smaller than this threshold tend to receive the highest payoffs.[37]

Another example of attack on a Bitcoin system (peer-to-peer network) is called an eclipse attack, which occurs in a context called a netsplit.[38] The first work to study eclipse attacks was that of Heilman et al[39] who demonstrated the first attack against Bitcoin's system by controlling hundreds of nodes and architectures as an unstructured random graph. In a P2P system, a distributed application architecture that divides tasks or workloads among peers, without stable hosts and peers, communicates through gossip protocols.[40] During an eclipse attack, a node becomes inaccessible to other nodes on the network, which means it can be used by an attacker. This is a simple way to attack P2P systems, including blockchains. The cost of launching an eclipse attack is high in an ideal state where each pair is always listening and talking to other peers. Therefore, the hacker must control the entire network to hack the P2P system. In practice, however, each pair can only exchange information with a small group of its peers. Thus, the cost to hack the system is not as high as first thought. In a simulation setting, Marus et al[41] used a botnet of 4600 IPs, with two IPs per group, and 5 hours of time to successfully fill a node's IP table. This is relatively small, because the Walowadac Botnet has 160 000 IPs and 25 000 groups, which shows that this attack is possible. The author performed a 1-hour live experiment using 400 IPs (400 groups), with one IP per group, and occupied 57% of the table tested with invader IPs.[39,41,42]

2.4.1 | Malleability attack

Let us assume that a dishonest user has n Bitcoin on an exchange and wishes to withdraw the coins. In this case, the user requests that the exchange send the Bitcoins to a specified address. This can automatically generate a transaction, which is transmitted for mining so that it can be included in the Bitcoin blockchain. However, the user can act as if the coins were not received, and use the transaction malleability flaw to reproduce the original transaction. He or she changes the digital signature to produce a different hash and retransmits the transaction with a different ID. There is a chance that the transaction will be confirmed on the blockchain first, which means the network will accept the transaction as valid. The dishonest user then complains about that exchange did not complete the transaction. When the exchange system checks the transaction and finds no record of its ID on the blockchain, it sends the coins again. According to Andrychowicz et al[43] this exchange is a custom implementation and is seemingly vulnerable to such attacks.

An attacker can cause extensive harm to the Bitcoin network by intentionally initiating transaction malleability attacks on multiple exchanges at once using software that is deliberately designed to create mutant transactions. The back-end accounting systems of these exchanges may be able to cope with moderate numbers of mutant transactions. However, if they are assaulted repeatedly, this can cause logistical problems.[43] In addition to protocol and network-level attacks, other attacks (eg, a Sybil attack) occur when networks lack admission controls. In this case, a blockchain network allows a single user to generate several online identities, which he/she uses to manipulate the consensus process. This allows the dominant group to confirm transactions and blocks, thus allowing double spending to occur. Dominance can also be achieved by other means, such as controlling 51% of the mining power on a PoW network.

2.4.2 | Real DOS attack against the Ethereum network

A DDoS attack (computational), which caused miners and nodes to spend a long time waiting to process blocks, affected the Ethereum network in September 2016. The root cause was EXTCODESIZE opcode—a low gas price opcode that mandated nodes to read state information stored on a disk. This opcode was initiated roughly 50 000 times per block by attackers. This lead to the network speed slowing down drastically.

In October 2016, a different DoS attack took place due to flaws inherent in the protocol. Attackers simply created large numbers of cheap, empty accounts in the Ethereum state through the use of SUICIDE opcodes that transferred

functionality to "poke" a new account into existence, and then repeatedly called it within a single transaction until it was deleted. Using this technique, 19 million empty accounts with zero balance and gas only valued at 90 were created by nodes from attackers. These ghost accounts can be turned into nonempty accounts by just sending Ethers of any amount to them. Technically, if a miner should accept transactions with zero fees, sending a transaction from an empty account is made possible. The only exception is when empty accounts need to be stored in an Ethereum state tree. Empty accounts cannot be stored in the state, which also means an attack cannot occur in the state. However, be it in the state or not, having a whopping 19 million empty accounts creates another issue—waste of hard drive space may increase sync time (especially fast sync and warp sync) and the reduce.[44]

2.5 | Specific flaw in DPoS

A while ago, we talked about PoW and PoS consensus security issues. In terms of security, there is a diverse opinion on which solution is the best. Both mechanisms have their own advantage and drawback for different types of networks, applications, and their overall security. Therefore, several consensus algorithms developed different blockchain implementation. Recently, delegated proof of stake (DPoS), which is a variant of PoS, became popular among the cryptocurrency market due to its high scalability.[45] DPoS consensus relay to a fixed number of elected entities called block producers (BPs) which are selected to create blocks in a round-robin order. BPs are voted into power by the users of the network, and each gets a number of votes proportional to the number of tokens they own on the network (their stake). Alternatively, voters can choose to delegate their stake to another voter, who will vote in the BPs election on their behalf. BPs are those responsible for creating and signing new blocks. They are limited in number and are elected by the voters. Block validators in DPoS refer to full nodes who verify that the blocks created by BPs follow the consensus rules. Any user is able to run a block validator and verify the network.[45,46]

In aspects of security, there are various concerns around DPoS, because algorithms provide a high level of scalability at the cost of limiting the number of block producers. Currently, well-known DPoS chains are EOS, BitShares, Steemit, Lisk, and Ark limited with 21, 101, 21 101, and 51 BPs, respectively.[45,47,48] According to Vitalik and Haseeb Qureshi, due to the nature of the permissionless network, an anonymous or pseudonymous party can participate in a combination of usage, validation, and block production. As a result, these networks cannot prevent Sybil attacks, where a single user creates multiple identities to use a network.[49] Since DPoS uses trusted witnesses, the chain is less vulnerable to the general flaw of proof of stake—"nothing-at-stake." However, DPoS remains at risk from BPs paying for votes or colluding for nefarious purposes (censorship attack). A group of dishonest users may also seek to take over the voting process. Stakeholder apathy encourages such attacks and remains a weakness in DPoS.[49] In this section, we outlined the major DPoS attack vectors (possible attack) and evaluated the threat they present.

2.5.1 | Block producers collude

Theoretically, in any blockchain system, the threat of BPs (miners, validator) collusion is looming. Because of DPoS relayed a small number of validators, it is always possible to organize collusion among them. According to Vitalik, three major attacks could launch in those colluding BPs: censorship, changing system parameters, and double spends attacks.[49]

Censorship attack

While a system is ultimately designed to encourage plutocracy and collusion among BPs, there is no guarantee for developers and users that their applications and transactions will not be censored. Censorship attack against to DPoS means that BPs refuse to process valid transactions. If only a single BPs (or minor group) censors an individual, it will not be a big issue for the network. The next honest majority of the BPs will probably validate the transaction in the following block, which could cause delayed transactions by not processing them in their blocks. In DPoS systems, the attack will be a success if the majority BPs is under the attacker's control (more than two-thirds of all BPs).[50]

Changing system parameters

Under DPOS, all changes must be triggered by active stakeholder approval.[48] It is technically possible that the BPs collude and change their protocol parameters unilaterally. If an attack is a success, then the attacker (or attacker group) may change the constitution, increasing their block rewards, forking out certain stakeholders, and other options on

protocol. The threshold for changing the rules is the same as replacing 51% of the elected witnesses. The more stakeholder participation in electing witnesses, the harder it becomes to change the rules. DPoS is designed in such a way that these attacks are not possible without implicit voter approval. In EOS case, changes to protocol parameters have time delays before they are actually incorporated. In addition, approval by 17/21 BPs is required to change the constitution, and they must maintain that approval for 30 consecutive days before the changes could take place.[47] In case the user does not accept changes, they can vote out that BPs during that time and replace them with producers that do not support the changes. Ultimately, changing the rules depend upon everyone on the network to upgrade their software, and no blockchain level protocol can enforce how rules are changed. This means that hard-forking "bug fixes" can be rolled out without requiring a vote of the stakeholders, so long as they remain true to the universally expected behaviour of the code. In practice, only security critical hard-forks should be implemented in such a manner. The developers and witnesses should wait for the stakeholders to approve even the most minor changes.[48,51]

2.5.2 | Exploit law voter turnout

This is one of the possible threats against a DPoS blockchain. In fact that on voting-based blockchain platforms, few participants actually show up and vote. On token voting platforms, users with small stake may not influence the direction of the platform with their votes. For small stake user, sometimes a vote may be costly than the value that voting brings. To solve this problem, DPoS allows proxy voting where the user can lend their voting power to another user. Results of these systems are that often overall voter turnout is low, and voting is mostly done by whales, exchanges, and wallet providers. Vitalik has explained risk as follows: Let us assume that 10% of the total supply of tokens was being used to vote, then a whale (or group of whales) with more than 5% of the total supply could step in and take over governance.[46]

2.5.3 | Attacks at scale

Another possible attack vector involves assumptions about what an industrial-scale DPoS blockchain looks like. According to Larimer, EOS is likely to scale in a way that large data centres act as BPs in order to provide the level of bandwidth and speed the network requires. It has not yet been observed in practice; however, if it happens, the implications are worth considering. If BPs are expected to be in dedicated data centres, it limits the number of potential BPs and especially limits the number of entities that could step in to replace BPs that are voted out. If there is not any BPs with sufficient resources to replace BPs that has been voted out, then, as a result, the network may suffer. Voters would have to decide between punishing a misbehaving BPs and lowering the overall resources of the network.[48,52]

3 | VULNERABILITIES ON BLOCKCHAIN 2.0 (ATTACKS AGAINST SMART CONTRACT)

Smart contracts on Ethereum represent the second generation of public blockchains, providing an open and global computing platform that allows for the exchange of cryptomeda (Ether). Smart contracts are intelligent, self-checking contract applications that provide a foundation for digital asset proprietorship and a range of decentralized applications in the blockchain domain. Ethereum and smart contracts are public, distributed, and immutable and, thus, are prone to vulnerabilities caused by simple coding errors. Ethereum is the most popular blockchain platform in terms of current cryptomeda market capitalization. Several studies provide taxonomies related to smart contract vulnerabilities. For example, Atzei et al[16] investigated security issues in Ethereum smart contracts and offered a taxonomy of common programming faults, which they refer to as vulnerabilities. They investigated 12 types of vulnerabilities, classified by Solidity, the Ethereum virtual machine (EVM), and Ethereum networks. Chen et al[53] examined underoptimized smart contracts, identifying seven gas-costly patterns within two groups: useless-code-related patterns and loop-related patterns. They found three representative patterns in 4240 real smart contracts, showing that 93.5%, 90.1%, and 80% of contracts are affected by each respective pattern.[53] Luu et al[54] examined the safety of running smart contracts based on Ethereum, finding several new security issues that could be exploited for profit. They created a symbolic execution tool called Oyente to find potential security bugs, enabling developers to write less vulnerable contracts. Table 1 (see also

TABLE 1 Smart contract vulnerabilities

Smart Contract	Vulnerabilities	Categories of Bugs
The DAO, Maker's ETH-backed token	Re-entrance	Re-entrance (recursive-calling vulnerability: A calling B calling A) Game-theoretic weaknesses
Rubixi, FirePonzi (Ponzi scheme)	Immutable bug Wrong constructor name	Variable/function naming mix-ups
King of the ether game	Out-of-gas send Exception disorder	Send failure due to 2300 gas limit
GovernMental (Ponzi scheme)	Immutable bug Stack overflow Unpredictable state Timestamp dependence	Arrays/loops and gas limits
FirePonzi	Type casts	Variable/function naming mix-ups
Parity multisig wallet	Visibility and delegate call	Unintended function exposure

Table 2 the appendix B) shows common pitfalls and vulnerabilities in smart contracts. Moreover, Vitalik et al[55] listed all major bugs found in smart contracts based on real attacks on Ethereum, which they classified into six categories.

3.1 | Re-entrance vulnerability (DAO attack)

Based on the largest attack (DAO attack) on Cryptomeo, re-entrance is the most severe vulnerability in smart contracts. The re-entry vulnerability is explained as follows. The atomicity and sequence of transactions may induce programmers to consider when a nonrecursive function is invoked because it cannot be reinserted before its completion. However, this is not always the case, because the fallback mechanism may allow an attacker to re-enter the caller's role. This can result in unexpected behaviour and possibly loops of invocations that eventually consume all available gas.[16] In June 2016, Ethereum suffered a DAO attack that led to a large-scale Ethereum node disruption, pitting the platform's developers against the unknown antagonists.[56] As a result, Ethereum implemented a hard fork, yielding two separate platforms, namely, Ethereum and Ethereum Classic.

In a DAO attack, the attacker first publishes a malicious smart contract, which is called the "splitDAO" function (this function updates user balances and totals at the end). In this way, the attacker sends 14 647 and receives 26 287 transactions, collecting Ether many times over within a single transaction. To prevent such attacks, the DAO community proposed a software fork with a NO ROLLBACK function, starting from block number 1760000. Here, any transactions that implement calls/callcodes/delegatecalls that execute code with an attacker hash code are rejected or rendered invalid.[57]

3.2 | Parity multisig wallet

In July 2017, the second biggest attack occurred when ETH was stolen on the Ethereum network. The problem reported by the Parity team that affected multisig portfolio contracts was part of the Parity software package. The attacker found a vulnerability in the 1.5+ version of Parity multisig wallet, resulting in the theft of more than 150 000 ETH (about $30 million). A parity attack expands a combination of unsafe visibility modifiers and misuses the calling of delegates with arbitrary data. Zeppelin Solution analyzed the cause and described the structure of the attack. An overview of the attack is provided below.

The attacker sends two transactions for each of the affected contracts, where the first transaction gains exclusive ownership of multisig, and the second moves all of the funds. The first transaction is a call to *initWallet* (line 216 of WalletLibrary), a function possibly created to extract the wallet builder logic to a separate library.[58] The portfolio contract forwards all unequal function calls to the library using *delegatecall* (line 424 of Wallet).[59] This makes it possible for all public functions to be called by any user with contract functions or inherited contract functions, or by external users from the library that includes *initWallet*, which can change contract owners. The attacker simply changes the status variable of the contract owners to a list containing only his/her address, requiring only a confirmation to execute a

transaction. Then, it is just a matter of invoking the second function to send all funds to an account controlled by the attacker. This execution is authorized automatically because the attacker is then the sole owner of multisig, effectively draining the contract of all funds.[60]

3.3 | King of the ether throne

King (of the Ether Throne) has not succeeded because of the high gas value necessary to send Ether to a contract address, where gas is the small amount of Ether required to perform computations on the blockchain. The flawed contract was a gambling scheme through which a user could buy the "crown" for a dynamic price. When a new "king" was crowned, the price for the crown would increase. Most payments expired to the leading king, with a small portion extracted for the contract owners. If the leading king used a contract address rather than a simple wallet to send its Ether and assume the crown, the contract would attempt to send the payment from the incoming king back to the ousted king's contract address. This should have been acceptable, because a contract can hold Ether. However, sending Ether to a contract costs more gas than sending it to a wallet. There were two problems with the scheme. First, the contract failed to check for the success of the transaction before continuing its execution. Second, it sometimes failed to allocate sufficient gas to send the Ether to a contract. In some cases, these meant payments would be reversed, but the contract would go on to crown the new king. This left the earnings from the previous king locked in the contract, accessible only by its owner.[61]

3.4 | GovernMental

GovernMental encounters a similar problem, albeit through subtler means. The contract was a Ponzi scheme. Users would send Ether to the contract with the promise of an increased return and with the chance to win a "jackpot." The jackpot was collected from a portion of each participant's entry and was awarded to the last user to join in the event that no one else had sent Ether to the contract in a 12-hour period.

The contract stored its users' addresses in a dynamically sized array and needed to iterate over the arrays in order to clear them when a jackpot was hit. However, it did not limit the size of the array. GovernMental eventually attracted enough users that the gas allocation could not cover the entire array. As a result, it would constantly fail to reset the game and award the jackpot to the winner, and the contract's state remained effectively frozen.

4 | VULNERABILITY ON BLOCKCHAIN 3.0

In this section, we will discuss existing security threats and their countermeasures for private blockchain and its underlying technologies. We provide a detailed discussion of potential vulnerabilities in the private blockchain network; we will be taking a close look at the broad attack vector and their impact on the particular components. There are some promising private blockchain implementations like Hyperledger, Corda, Quorom, Exonum, Ethereum as well as Quorom and Ethereum private use in Ethereum public blockchains, which can be forked and redeveloped for the use in private blockchain.[62] Corda was designed for financial-grade distributed ledger technology (DLT) platform. However, Corda designed for semiprivate networks requires obtaining an identity signed by a root authority for admission.[63] Thus, we will approach Hyperledger private blockchain from a security perspective. The section will also provide a general summary of possible risks on private blockchains.

4.1 | Attack against Hyperledger Fabric

Over time, these cyberattack incidents resulted in enhancements being made to BT. Hyperledger was created in response to business needs, with the BT transforming its framework structure. This section discusses the security design and vulnerabilities of the current framework. Hyperledger is an open-source blockchain project created by the Linux Foundation to back the cooperative development of blockchain-based distributed ledgers. The project's objective is to help businesses usher in a new era of trust, transparency, and accountability, with a particular focus on improving the performance and reliability of distributed ledgers.[64-66] The Hyperledger Modular Umbrella approach includes the

following five frameworks, each of which provides its own advanced functions in order to cope with business and industry requirements: Fabric, Burrow, Iroha, Sawtooth, and Indy.[66,67]

IBM operates a smart contract (chaincode) in Fabric, and developed Hyperledger Fabric as a Hyperledger project. The modular architecture delivers high degrees of performance, scalability, and levels of trust. In addition, Hyperledger Fabric is a framework for permission networks. All participants have known identities, and every user participating in a transaction must register on the network in order to obtain an enrollment ID. Fabric supports pluggable implementations of a function that allows developers to use any programming language to implement chaincodes, with the most common being the Go language, run within Docker containers.[65] The intention of the Hyperledger Fabric is to offer a number of SDKs for a wide variety of programming languages, including Node.js and Java SDKs. The Hyperledger Fabric SDK for Node.js is designed in an object-oriented programming style. Its modular construction enables application developers to plug in alternative implementations of key functions, such as crypto suites, the state persistence store, and a logging utility.[68] Node.js has specific security flaws because it is susceptible to a remote DoS attack.[69] Node.js and Go can be exploited to execute code remotely, resulting in what is called a DNS rebinding vulnerability.[70] The attack is possible from a malicious website that accesses the web browser on a computer that has network access to the computer running the Node.js or Go process. The malicious website can use a DNS rebinding attack to trick the web browser and bypass same-origin-policy checks, allowing HTTP connections to the localhost or to a host on the local network. If a process with an active debug port is running on the localhost or on a host on the local network, the malicious website can connect to it as a debugger and get full access to the code execution. In order to assess the overall security of Fabric software products, Graham et al[71] performed penetration testing on Hyperledger. Based on the risk profile, primary security concerns, and vulnerabilities identified at the point of the engagement, they found that Fabric requires moderate attention. Table 2 shows the overall security position of Fabric 1.1 software products, based on the technical report.

4.1.1 | Hyperledger security design (insufficient chaincode sandboxing)

Fabric chaincode runs in a secured Docker container that is isolated from the endorsing peer process. However, this is insufficient to prevent malicious chaincode from being written. The main concern is that chaincode is accessible on the network and can easily be downloaded and installed in other software packages and security tools. Moreover, it can run for long periods. The use of Docker greatly constrains what the chaincode can do. However, the chaincode still has sufficient freedom, given that it can do the following:

- Install arbitrary software within the container, including security tools such as Nmap.
- Perform port scans against public or private networks that are visible to the node.
- Exploit any vulnerable hosts that are discovered.
- Accept commands from, and exfiltrate results to, a remote command-and-control server.
- Continue executing for a long period (perhaps indefinitely).

Remote access trojan (RAT) malware is formed by bringing together these capabilities, creating a foothold in a corporate network that allows other systems to be scanned and attacked. The installation of malicious chaincode would be a nontrivial exercise for most threat actors, given the level of access required. However, plausible scenarios exist. For example, an attacker may create a new ledger with associated malicious chaincode, influence others to participate in or infiltrate the organization responsible for developing and maintaining the chaincode for an existing ledger, and then publish an update. Note that the chaincode does not necessarily contain any overtly malicious functionality at the time it is installed on the network. It merely needs to be able to download and execute code from a command-and-control server at some future point in time.[71]

4.1.2 | Docker container security

Docker containers are, by default, quite secure, especially when running processes as nonprivileged users inside the container. While we certainly need to be aware of the issues related to using containers safely, if used properly, they can provide a more secure and efficient system than virtual machines used alone. In this section, we discuss the

TABLE 2 Software security assessment of Hyperledger v1.1

Description	Adverse Effects	Recommendation	Security Level
Chaincode sandboxing insufficient to prevent malicious behavior	With malicious chaincode capable of performing an Nmap scan, an attacker may connect to a server (command-and-control), that provides sufficient functionality for a "remote access Trojan" (RAT) to be implemented as chaincode. If installed on an internal company network, such a trojan would provide an excellent foothold that an attacker could use to pivot to other systems.	1. Restrict network access provided by the Docker container, preferably to just the node that created it; 2. Have the chaincode execute as a nonroot user; 3. Limit the length of time for which chaincode can run, and ensure that the chaincode cannot achieve persistence by other means (eg, by spawning a subprocess or running as a cron job).	Chaincode sandboxing insufficient to prevent malicious behavior
Comment headers insufficient for checking implementation and usage	Comment headers provide a detailed specification of the behavior of each function and it is desirable that the "contract" between each function and its callers be documented, because this would limits the volume of code. This is a security risk if it makes the code-review process less effective.	Specify function interfaces in comment headers	Comment headers insufficient for checking implementation and usage
Log injection	1. Fabricate log messages; 2. Corrupt the log to prevent it from being processed automatically; 3. Exploit terminal emulators or other software used to view the log (uncommon on modern systems)	Escape untrusted strings before logging	Log injection
Code injection	Code injection or remote code execution (RCE) is an attack type that happens by the injection of code which is then interpreted/executed by the application. If it is possible to embed newline characters within a string, then it is possible to start a new command on a new line, allowing access to the full range of possible commands.	Either document behavior or validate arguments	Code injection
Remote imports allowed/encouraged in chaincode	One of the more notable features of the go programming languages is the ability to import packages from a remote repository, identified by a URL. Fabric encourages the use of this feature in chaincode, being the method used in the documentation to load the shim package and other support code	Require whitelisting of remote repositories	Remote imports allowed/encouraged in chaincode

Source: Graham Shaw (2017).[53]

potential security issues related to containers, as well as the major tools and techniques available for securing container-based systems.

4.1.3 | Container resource abuse

Containers are more numerous than virtual machines, on average. In addition, because they are so lightweight, large clusters can be spawned on modest hardware. While this is definitely an advantage, it implies that a lot of software entities are competing for host resources. Software bugs, design miscalculations, or malware can easily result in a DoS attack if resources are not properly configured, because all containers share kernel resources. If one container can monopolize access to certain resources—for example, memory, or more esoteric resources, such as user IDs (UIDs)—it can starve out other containers on the host, resulting in DoS, whereby legitimate users are unable to access part or

all of the system. In order to prevent this kind of attack, it should be possible to limit the resources allocated to each container. Cgroups are the key components that Docker employs to deal with this issue. They control the amount of resources (eg, CPU, memory, disk I/O) that any Docker container can use, ensuring that each container obtains its fair share of the resources and preventing any container from consuming all resources. They also allow Docker to configure the limits and constraints related to the resources allocated to each container. For example, one such constraint limits the CPUs available to a specific container.[72,73]

4.1.4 | Container breakouts

Through the Docker container, an attacker can gain various privileges or can bypass isolation checks, thus accessing sensitive information from the host. Normally, it should not be possible for an attacker to gain access to other containers or the host. This is because Docker always runs in the background when a container is started. In this way, Docker creates a series of *namespaces* and control groups for the container. The role of the namespaces is to isolate processes running inside the container. However, they cannot carry out this function outside the container in which they are working. A notable flaw of this process is that users are not *namespaced*. Therefore, any process that breaks out of a container will have the same privileges on the host as it did in the container. For example, if a user is root user in the container, then it will be a root user on the host. In addition, by default, the Docker daemon runs as a root. This can cause potential elevation attacks (elevated privileges gained by user), such as those of the root user, usually through a bug in the application code that needs to run with additional privileges. If the host system is configured correctly, reducing the container's root capabilities and creating a user-level namespace, then containers can interact with each other through their respective network interfaces, just like they can interact with external hosts. In addition, accessing a database or service may reveal certain details, such as an API key or username and password. An attacker who accesses this information will also have access to the service.[72,73]

4.2 | General risk on private blockchain implementation

In terms of security, there are certainly advantages of private blockchains where the miners or validators cannot be anonymous. Because organization preselects the participants and thus are highly trusted. Therefore, the chances of someone acting maliciously on a network are less. Additional privacy solution is more assured on a private blockchain than public.[74] Different business cases have different requirements, and no solution would fit all the cases. Understanding the different properties of platforms is vital to in making the decision regarding blockchain platform and even evaluating whether using blockchain to implement a business case make sense. If high confidentiality, transaction throughput, and immediate finality are required, then private permissioned blockchains seem a promising technology. We highlighted below some of the possible security issues and risks of private blockchain implementation:

- Poor architecture design: Private ledger network may get split and replicated into parallel fork chains creating ambiguity among child blocks for parallel parent fork chains. This replication may be susceptible to several attacks in parallel fork chains.
- Poor network design: Ledgers may be susceptible to DoS or transaction spamming if proper Membership Service Providers[75] (identity management) is not set for the private blockchain.
- Poor cryptography: Enterprise blockchains need to encrypt data to ensure maximum security. Apart from that, it needs to follow the cryptographic standards that apply to public blockchains to secure transaction messages and ensure transaction authenticity. If cryptographic keys are not stored or maintained properly, it could cause the compromise and disclosure of private keys leading to fraudulent transactions or loss of assets. This will lead to the compromising of the integrity and privacy of the operations.
- Poor access management on smart contracts: Basically smart contract security depends on codified business logic. For example, using Hyperledger Fabric needs codified business logic in Golang or JavaScript, which you can then wrap around the nodes responsible for this sort of logic with Docker. With Exonum, you need to define all your business logic in the Rust programming language and compile it as a module in you blockchain node software.[76] In Quorum (Ethereum private also), it needs Solidity, which will run on a virtual machine on each blockchain node. That is why code vulnerabilities like call stack, stack size limit, reentrancy, malicious libraries, type casts, and open sensitive information might deviate smart contracts' intended behaviour to malicious transactions or even unintended take over for further compromise.

- Consensus: Enterprise blockchain transactions are more complex than public transaction. The consensus mechanism would need to involve different roles, for example, the need to work on different parts of the transaction lifecycle, such as endorsement, ordering, and validation. Mainly private blockchain uses different variants of voting-based BFT algorithms (pBFT, rBFT, aBFT) and Proof-of-Authority available on Ethereum private. Generally, BFT-based consensus suffers byzantine fault tolerance. In addition, some form of BFT-based consensus needs the nodes in the network to be known and must be totally connected.[64]

5 | POSSIBLE COUNTERMEASURES TO VULNERABILITIES

Here, we describe the existing countermeasures and detection algorithms available to BT that can be used to ensure privacy and security. For a comprehensive overview of this topic, we extract several papers and internet resources from scientific databases. Figure 4 summarizes the results, including cutting-edge solutions applied to blockchain environments that address security threats and enable strong privacy.

5.1 | Slasher: A punitive PoS algorithm

The Slasher technique is a hybrid of the PoW/PoS algorithms, and was first described by Vitalik Buterin, one of the architects of Ethereum. Slasher uses PoW to generate and mine blocks. However, each unit is validated by both algorithms. On the other hand, there is some modification. During the creation of a new block (block K), a miner must include the value $H(n)$ for some random n generated by the miner. This number serves as proof of mining. The miner can claim his/her reward for the block, creating a special transaction that opens n. The reward for mining is closed for 100 blocks and is limited in time. As such, the miner can claim the reward by releasing a transaction and uncovering n block $K + 100$, $K + 101$, ..., $K + 900$, where the average time taken to create a block set is 30 seconds. To create a valid block, it must be cryptographically signed. Signatory users are selected randomly using a condition. Let us assume that M is the total money supply and $n[i]$ denotes the n value of block i. At block $K + 1000$, an address A with balances B ($B = \sum a\ bal(a)$—total number of coins in circulation) used to sign a privilege condition looks as follows:

$$\text{signers(h)} = \left\{A : hash(n(K-2000), n(K-1999), \cdots, n(K-1901), A) \leq 64M\ bal\left(\frac{A}{B}\right)\right\}.$$

This means that an address has the chance of obtaining a signing privilege proportional to the amount of money it has and, on average, 64 signing privileges are assigned to each block. To determine the correct block for the fork, the total number of signatures is used. By signing the block, the user receives a reward that is closed for the next 1000 blocks. The reward for the signature is higher than that for mining, which prevents an arms race among the miners. To combat forks of the blockade, Slasher uses a mechanism similar to that of Tendermint. Figure 5 shows how voters that vote on the wrong fork are penalized and how voters that double-vote is presented.

If the user of the system observes several blocks of the same height, signed by the same certifier, the user can publish a transaction with these two signatures. If this transaction is included in the block before the reward for the block becomes available for use by an unscrupulous certifier, 33% of the reward is paid to the user who has retreated from the protocol, and the remaining funds are destroyed. Slasher largely prevents short-term block forks. In fact, the vulnerability of PoS to short-term attacks stems from the following observation. In the PoS architecture, the probabilities of creating blocks for competing chains are independent of each other, because they depend on the hash of the last target block. For system users, it makes sense to try to create blocks based on each of the chains, because this increases the expected reward. On the other hand, in Slasher, the probability of becoming a block certifier is determined using a large time interval and is the same for all chains, assuming they are relatively small. Thus, users have no reason to support the fork of the block, because they know in advance whether they will sign the blocks in future. For users, Slasher does not support multiple branches of the blocking system if they are not confident that the fork will survive more than 2000 blocks. Because Slasher uses PoW to create blocks, long-term attacks require significant computing power. This gives Slasher a significant advantage over PoS architectures that do not use PoW (eg, BitShares).[77]

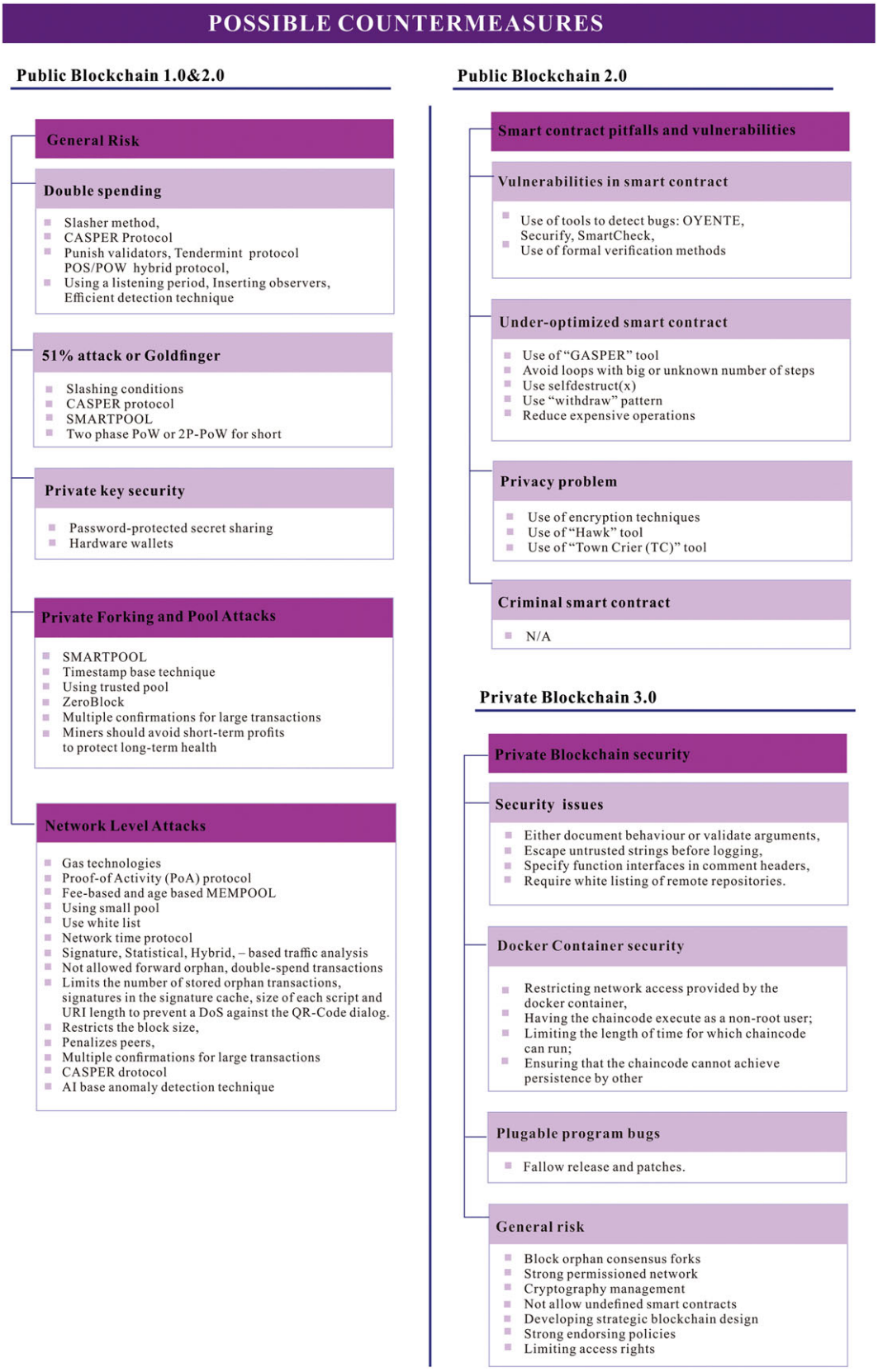

FIGURE 4 Possible countermeasures for vulnerabilities

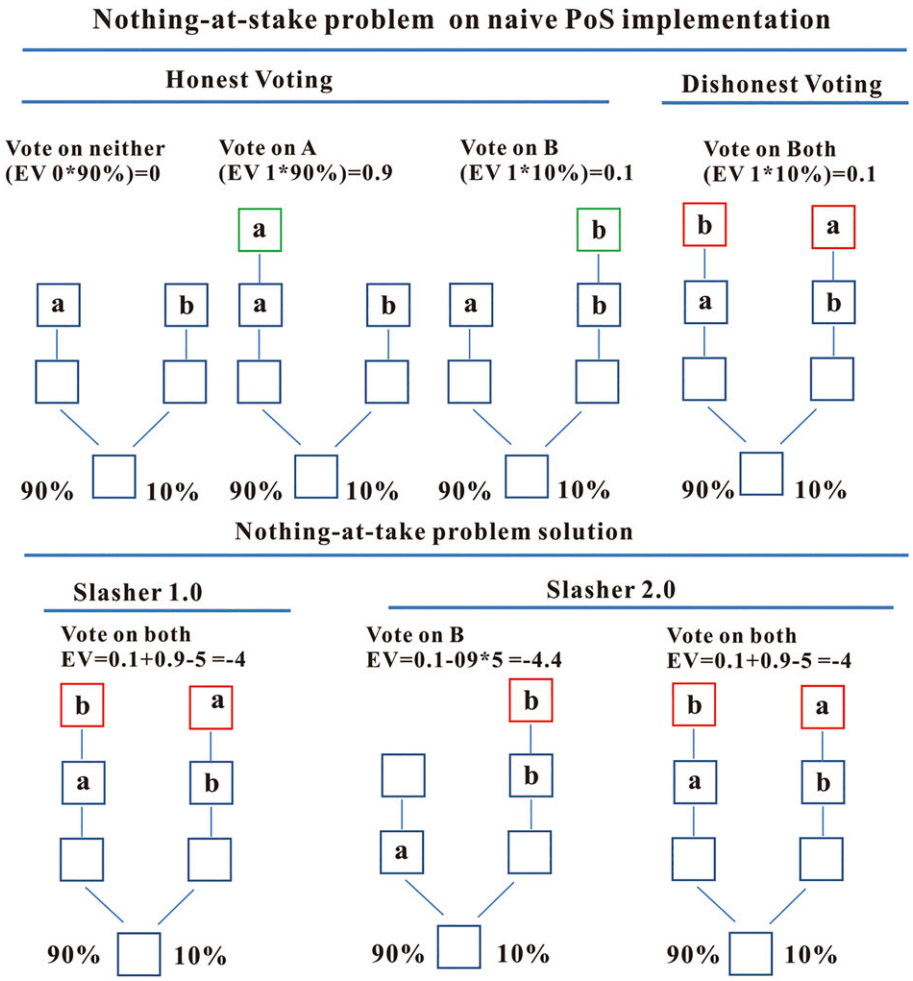

FIGURE 5 Nothing-at-stake problem with Slasher solution

5.2 | CASPER protocol

Several cryptocurrency consensus algorithms assign rewards to certifiers who participate in the PoS consensus algorithm. From an algorithmic perspective, there are two major types of PoS: chain-based PoS and Byzantine fault tolerance (BFT). The algorithm for BFT-style PoS allows certifiers to "vote" with finality conditions and slashing conditions by sending one or more types of signed messages. Vitalik et al[78] introduced CASPER, a PoS-based finality system that followed the BFT tradition, with some modifications. CASPER is a partial-consensus mechanism (a hybrid PoW/PoS system designed to punish all malicious elements) that combines PoS algorithm research and BFT consensus theory. CASPER introduces several new features, some of which are described below.

5.2.1 | Accountability

CASPER uses smart contracts to track the stakes and funds created. If a certifier violates a rule, CASPER can detect the violation and identify the certifier, allowing it to penalize dishonest certifiers and, thus, solve the "nothing-at-stake" problem in chain-based PoS. CASPER provides a partial solution to the nothing-at-stake problem by punishing stakers. In addition, it provides a full solution to the problem using the punishment feature and allowing users to choose off-chain centralized servers to determine which blockchain is legitimate in the case of multiple options. The penalty for violating a rule is the "deposit" required by CASPER, which is locked for a set amount of time in order to participate in staking. CASPER's accountability means that if two conflicting checkpoints (HASH1 and HASH2) are finalized, then at least 1/3 of all certifiers must have violated some slashing condition (which means at least one-third of the total deposit is lost).

5.2.2 | Dynamic certifiers

The set of certifiers needs to be able to change. New certifiers must be able to join, and existing certifiers must be able to leave. This feature allows CASPER to defend itself against well-known PoS attacks: long-range revisions and catastrophic crashes.[78]

5.3 | Tendermint

Tendermint proposed the concept of blocking, in which security is provided by a modified reconciliation protocol based on share confirmation. Each block must be cryptographically signed by certifiers in the Tendermint consensus protocol, where certifiers are simply users who confirm their interest in the security of the system by closing their funds with the help of a bonding transaction. The weight of the opinion of each certifier is proportional to the number of closed funds. After serving as a certifier, the user gets access to the closed funds by the return of the pledge (unbonding transaction). The funds are unlocked with a certain delay (unbonding period). A block is considered correct if it is signed by certifiers that, in sum, have at least two-thirds of the total weight of votes. Thus, a fork of a blockage is possible only if there is a group of certifiers with at least one-third of the votes that sign the blocks in both competing blockhouses. In the case of fork certifiers, signing several blocks can be punished by publishing an arbitrary transaction with a certificate containing evidence of their malicious intent, for example, their digital signatures for blocks of the same height from both chains. The transaction certificate destroys all mortgaged funds of certifiers acting outside the protocol. This logic prevents short-term attacks. However, long-term attacks are still possible. For example, certifiers with two-thirds of all votes can conspire and publish a fork of the blockade after their funds are unlocked. To prevent long-term attacks, a mechanism can be used that prohibits long forks of blocking (such as Nxt).[79]

5.4 | Smart pool

Mining pools are based on a centralized system in which individual miners join a pool and mine blocks to solve puzzles by combining their resources to increase their computing power. After successful mining, the reward is split between the miners' based on their individual contributions, thereby allowing miners to have a stable income. For this purpose, a great number of mining pools have been created, and research in the field of miner strategies is also evolving. However, centralized cryptocurrency mining pools create problems. For example, five Bitcoin mining pools control 70% of the total hash power, and two Ethereum pools control over 50% of its total hash power. This represents a serious problem, because blockchains should be decentralized in nature.[80] Due to the dominant role played by such mining pools, attack vectors that exploit vulnerabilities in pool-based mining have also increased, making blockchain network architectures vulnerable to DDoS attacks.

As a solution, Loi et al[80] proposed SMARTPOOL, a novel protocol design for a decentralized mining pool. The authors focus on two key contributions. First, they propose running a decentralized pool mining protocol as a smart contract on the Ethereum network. Second, their protocol is efficient and can scale to a large number of participants. SMARTPOOL maintains two main lists in its contract state: a claim list *claimList* and a verified claim list *verClaimList*. When a miner submits a set of shares as a claim for the current Ethereum block, it is added to the *claimList*. This step acts as a cryptographic commitment to the set of shares claimed to be found by the miner. Each claim specifies the number of shares the miner claims to have found and each has a structure that aids verification in the subsequent step. SMARTPOOL then verifies the validity of the claim. Once verified, it moves it to the *verClaimList* list. Claim verification and payments for verified claims happen atomically in a single Ethereum transaction. Each claim allows miners to submit a batch of shares (eg, one million shares). Submitted claims need to include sufficient meta-data for verification purposes.

During the first step of mining the shares, if a miner finds a valid block in the target cryptocurrency, it can directly submit the block it found with a SMARTPOOL address as the beneficiary. Thus, miners receive payouts for their shares of one or more blocks after SMARTPOOL receives a reward from the target network. This mechanism ensures that the cryptographic commitment strictly precedes the verification step. Pools have several benefits, such as omitting the centralized operator that operates the protocol and manages other participants. This simply means that the pool is a fully decentralized protocol with low running costs, offering participants a comparable reward for communication expenses, communication bandwidth, local computation, and other costs. Furthermore, the protocol protects participants from

attackers who might steal rewards or prevent others from joining the protocol. This protocol is fair because participants receive rewards in proportion to the share of their contributions.[80]

5.5 | DoS/DDoS and SPAM attack prevention

DoS/DDoS and SPAM attacks are still the biggest threats to centralized systems. However, a decentralized system can also fall victim to such attacks, and thus, both systems need solutions. In accordance with security policy rules, if a DoS or DDoS attack is detected, it needs to be registered for further auditing. Services that are not related to security can also be used to detect a DDoS attack, for example, those services that redirect traffic over other communication channels, including backup servers that can be used for copying information. Thus, there are various ways in which DDoS attacks can be detected and prevented that are largely dependent on the system of protection. Countermeasures for DDoS attacks can be divided into preventive and reactive defence strategies. To initiate a preventive defence, secured computers can be used to reduce the possibility of attack, but they cannot guarantee perfect efficiency, even though they can reduce the frequency and strength of DDoS attacks. In a reactive defence strategy, early warning systems are used to detect attacks and react accordingly. Here, the main detection strategies are a signature based on a qualitative traffic analysis, an anomaly based on a quantitative traffic analysis, and a hybrid strategy that combines the merits of the first two methods. Today, anomaly detection methods have become popular and are widely used for detecting DoS/DDoS and SPAM attacks. Here, the parameters of observed network traffic are compared with normal traffic, and new attacks are detected to prevent a false alarm. The model of "normal traffic" must be kept updated, and the threshold that determines an anomaly must be properly adjusted. Table 3 summarizes the major detection approaches and their advantages and disadvantages.

Blockchain networks show strong resistance to DDoS attacks. In general, each distributed node contains data, making it inefficient and unreasonable for a hacker to attack the network. However, blockchain-based systems are not completely free from DoS/DDoS attacks. In order to effectively combat DoS/DDoS attacks, a user needs to know the type and characteristics of the attack and must inform security services promptly to obtain such information. Making adjustments to the system helps, but it is difficult to determine whether an attack is performed by at malicious programmer or as a DoS from unauthorized interference. Flooding a node with large amounts of junk data may make it so busy that normal transactions cannot be processed. Bitcoin includes some protection against DoS attacks, but it is still vulnerable to more sophisticated DoS attacks. Table 4 shows the current Bitcoin protocol rules used to defend against a DoS attack.

Furthermore, the Bitcoin protocol version 0.7.0 for client protection does not allow the forwarding of orphan transactions or blocks, double-spend transactions, or sending the same block transaction or alert to a peer more than once. The unspent transaction output set is only stored in memory, and the remaining data are stored on disk. However, Bitcoin clients do not directly limit peer bandwidth or CPU usage.[81] All of the above techniques have disadvantages. Thus, several studies have proposed new techniques for detecting DDoS/DoS or SPAM transactions in an anonymous environment. According to Muhammad et al[82] in the memory pools of cryptocurrency networks, DDoS attacks can lead to massive transaction backlogs and higher mining fees. The authors also propose reactive measures to protect against the same attack. These measures include fee-based and age-based designs, which can optimize the mempool size and help to counter the effects of DDoS attacks. In the fee-based design, an incoming transaction is accepted by the mempool if it pays both the minimum relay fee and the minimum mining fee. The key idea behind this scheme is to counter the strategy of an attacker by only accepting transactions that will be mined into the blockchain. As a result, this technique puts a cap on incoming transactions and filters spam transactions, thus reducing the mempool size. In an age-based design, the authors counted the number of inputs or parent transactions for each incoming transaction, and initialized a variable called "average age" to zero. Subsequently, they calculated the average age of the transaction by adding the age of each parent transaction and dividing by the total number of parent transactions. This gives an estimate of a mean confirmation score of the incoming transaction. After applying this filter to the mempool, the "minimum age limit" could take any arbitrary value greater than zero. According to Bitcoin, a confirmation score of six is considered sufficient for a transaction. If the transaction's mean age value fulfils the age criterion, only then does the mempool accept the transaction. In this way, all unconfirmed transactions generated by an attacker or Sybil nodes will be rejected by the mempool, while the transactions of legitimate users will be accepted. If an attacker still wants to spam the network, it may return all its transactions as mined and wait for them to acquire a significant age. This increases the cost of the attack and reduces the time window within which the attack can be launched.[82]

TABLE 3 Major detection approaches and their advantages and disadvantages

Approaches	Advantage	Disadvantage
Signature-based or knowledge-based detection	The patterns of anomalies are filed in a directory or database. Useful in recognizing only defined and well-known attacks in real-time and performs in supervised mode. Deployed on any network (source-end, victim-end, or core-end network). Offers the features of robustness, scalability, and flexibility.	Unable to identify new or novel attacks. Prior knowledge of attack signatures is needed, which needs to be updated regularly. Moreover, detailed knowledge is needed for better detection. This is a highly complex and prolonged task, because the effectiveness of the knowledgebase requires thoughtful and comprehensive dissection of each vulnerability. This approach faces various generalization issues.
Anomaly-based detection	Identifies an attack by labeling the activity as either abnormal or normal. Able to expose new, unusual, or "zero-days" attacks and exceptional patterns if they do not correspond to the presumed normal functionalities. Profiles of normal activities can be customized for systems, applications, or networks.	Difficult to extract network features. Thus, a training phase is needed to recognize a normal activity profile. Setting a threshold value to prevent false positives and false negatives is difficult. Defining a rule set is difficult. High false positive rate leads to low detection efficiency. Offers low throughput and is computationally expensive because of the cost of retaining a record of, and perhaps refreshing several system profile metrics.
Hybrid detection	It runs in supervised and unsupervised mode. Combines two or more detection approaches. While it still misses some attacks, its low false notification rate strengthens the plausibility of exploring most alerts.	Exploits the benefits of both signature and anomaly-based detection procedures, so the resulting hybrid systems are not perpetually favorable. Leads to high complexity and implementation costs.

Source: Parneet Kaur (2017).[61]

TABLE 4 Bitcoin protocol rules built to prevent DoS attack

	Protocol Rules	Limitation
1	Bans IP addresses that misbehave	24 hours default
2	Limits the number of stored orphan transactions	10 000 by default
3	Limits the number of stored signatures in the signature cache	50 000 signatures by default
4	Considers nonstandard signature scripts	If size greater than 500 bytes
5	Restricts the block size	1 megabyte
6	Limits the size of each script	Up to 10 000 bytes
7	Limits the number of key arguments OP_CHECKMULTISIG	Up to 20 keys
8	Finite number of stack elements that can be stored simultaneously	Up to 1000 elements
9	Limits the number of signature checks a block may request	Up to 20 000 checks
10	Considered nonstandard transaction	Transactions greater than 100 kilobytes

5.6 | Gas technologies

Ethereum uses the gas mechanism to ensure that every operation in smart contracts running on the EVM will eventually terminate and to broadcast and confirm transactions on the network. The Ethereum protocol charges a fee for each computational step in a transaction. Every transaction is required to include a gas limit and a gas fee that the miner is willing; miners have the choice of including the transaction. The gas price per transaction or contract is set up to deal with the Turing-complete nature of Ethereum and its EVM code to prevent infinite loops. If there is not enough Ether in the account to perform the transaction or message, then it is considered invalid. The idea is to stop DoS attacks from infinite loops, encourage efficiency in the code, and to make an attacker pay for the resources they use, including computation, bandwidth, and storage. However, setting the gas costs of EVM operations improperly allows attackers to launch DoS attacks on Ethereum. On the other hand, it is difficult to properly set the gas cost of each operation because this requires a deep understanding of EVM internals, as well as a correct measurement of resource consumptions by EVM operations on different types of computing resources. As mentioned earlier, in 2016, two DoS attacks were identified that exploited such operations, repeatedly executing two underpriced operations, namely EXTCODESIZE and SUICIDE, thus resulting in slow transaction processing, wasted hard drive space, and long synchronization times. Therefore, attackers can launch DoS attacks on Ethereum at a low cost by exploiting underpriced operations.[83]

5.7 | Countermeasure methods and tools for specific smart contract vulnerabilities

This section provides an overview of smart contract privacy and analysis tools that help developers to write a correct smart contract. This section also covers current techniques and methods used to detect vulnerabilities, along with their architectural classifications for smart contracts. We provide a detailed overview of the tools (Oyente, Securify, SmartCheck, GASPER, Hawk, and Town Crier (TC)) and their specifications and limitations.

The first security analysis and symbolic execution tool proposed by Luu et al[54] was called Oyente. It was meant to find potential risk/threats (security bugs), which included transaction-ordering dependence, timestamp dependence, mishandled exceptions, and re-entrance on the existing Ethereum system. Furthermore, it could analyze both Solidity and the bytecode of a smart contract, and it had a modular design consisting of four main components, namely CFGBuilder, Explorer, CoreAnalysis, and Validator. Each component had significance, for example, CFGBuilder was responsible for constructing a control flow graph (CFG) of the contract, where nodes were basic execution blocks and edges signified the execution jumps between the blocks. Explorer was the main module that systematically executed the contract. Its output was then inputted to CoreAnalysis (location of implemented concepts to target the vulnerabilities). Finally, Validator was responsible for filtering out false positives before reports were issued to the user. It is important to note that symbolic execution can achieve better precision (or fewer false positives) compared with traditional approaches based on a static taint analysis or a general data flow analysis.

In the latter approaches, abstract program states are sometimes combined and real-life execution of admitting states never occurs, leading to a high false positive rate.[54] Based on the above explanation Figure 6 demonstrates the architecture and execution process of OYNETE.

An alternative security scanner for Ethereum is Securify, which exists as a web-based security analysis tool that delivers automation (to enable everyone to verify smart contracts), guarantees (for finding specific vulnerabilities),

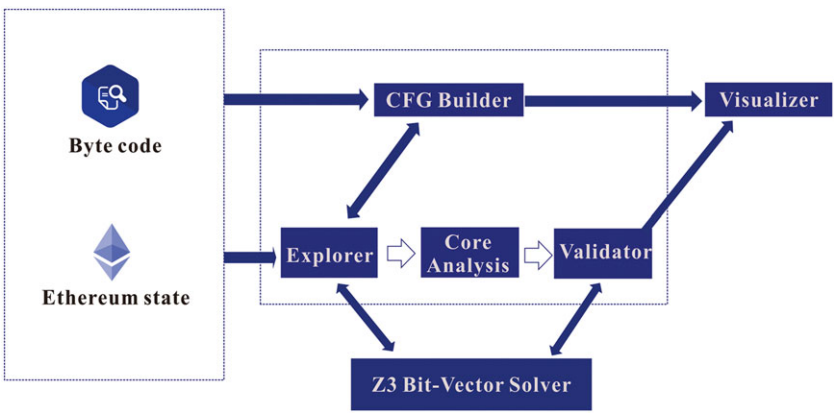

FIGURE 6 OYNETE architecture and execution process

and extensibility (to capture newly discovered vulnerabilities). Although it uses formal verification, it still relies on static analysis checks. It has the multifaceted role of analyzing bytecode and Solidity, plus analyzing smart contracts using their addresses.[84]

Another web-based security code and extensible static analysis tool is called SmartCheck. It automatically searches for bad coding habits and vulnerabilities. Moreover, it gives explanations for specific vulnerabilities and suggests possible solutions to avoid these security issues. However, it only runs on Solidity code, and its specific methods for identifying vulnerabilities (eg, symbolic execution, formal verification, etc.) have not been confirmed. The correlation of each detected vulnerability is shown alongside its severity level. Identified vulnerabilities are identified as DOS by the external contracts, gas-costly patterns, locked money, re-entrance, timestamp dependency, tx.origin usage, and unchecked external calls, while low severity (warning level) vulnerabilities such as compiler version not fixed, style guide violations, and redundant functions can also be identified. However, detecting high-grade bugs such as taint analysis, or even manual audit is a major flaw of the SmartCheck.[85]

Also, there is another security tool called GASPER by Chen et al.[53] It tackles the vulnerability in under-optimized smart contracts that consume more gas than required. By analyzing the smart contract bytecode, it can identify gas-costly patterns by conducting symbolic executions of bytecodes to cover all reachable code blocks (a block is a straight-line code sequence with no incoming branches except for the entry, and no outgoing branches except for the exit). The GASPER analysis process is as follows: First, it disassembles the bytecodes of the smart contracts it analyzes using disasm (provided by Ethereum). Second, it constructs a CFG. The CFG efficiency improves gradually during symbolic execution if new control flow transfers are found. This execution starts from the root node of the CFG and traverses the CFG. While this occurs, if GASPER faces a conditional jump, it uses the Z3 solver to check which branches are possible. If there are two possibilities, it selects one branch following an in-depth search. Although it is limited to gas-costly pattern bugs, it is still a work in progress.[53]

So far, we have reviewed security tools that have helped to identify the technical problems that can be related to bugs and vulnerabilities. However, for a blockchain system (which suffers from transaction privacy problems) and smart contracts to be useful, privacy is needed. One method proposed by Ahmet et al[86] is called Hawk. It is a framework for building privacy and preserving smart contracts and does not require the implementation of cryptography; the compiler automatically generates an efficient cryptographic protocol that uses the concept of zero-knowledge proofs. When this tool is used to develop smart contracts, the contract contains both private and public portions together. While the private portion relies upon the private input data and the financial function related code, the public portion contains data that does not touch private data or money. In general, the Hawk system shows how secure computations can be implemented on top of a public system such as the blockchain.[86]

An authenticated data feed system Town Crier which enables smart contracts to use data from outside the blockchain while still ensuring the preservation of anonymity using encrypted parameters was presented by Zhang et al.[87] As Figure 7 shows, its main role is to act as a connector between smart contracts and existing HTTPS-enabled websites that are trusted for blockchain applications like Ethereum. Also, it uses a front-end smart-contract with

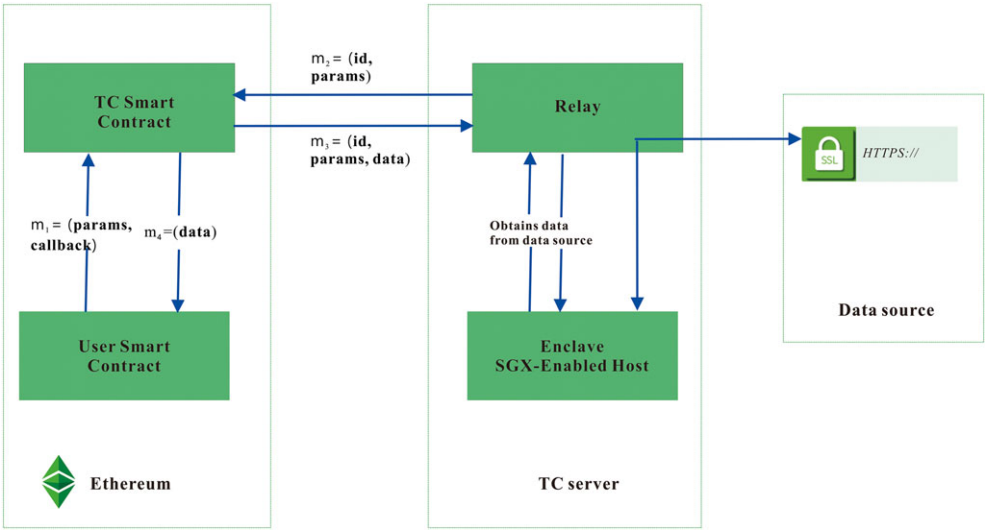

FIGURE 7 Town crier architecture and data flow process

TABLE 5 Vulnerability detection tools and analysis results

	Vulnerability Detection Technique					
Vulnerabilities	**OYNETE**	**SmartCheck**	**Security**	**Gasper**	**Hawk**	**Town Crier**
Timestamp dependency	+	+	−	−	−	−
Transaction-ordering dependence	+	+	+	−	−	−
Re-entrance vulnerability	+	+	+	−	−	−
Callstack depth attack	+	+	+	−	−	−
Mishandled exceptions (exception disorders, unchecked-send bug, and gasless send)	+	+	+	−	−	−
Tx.Origin usage	−	+	+	−	−	−
Gas costly patterns	−	+	−	+	−	−
Blockhash usage	−	+	−	−	−	−
DoS by external contract	−	+	−	−	−	−
Privacy issue	−	−	−	−	+	+

dependable hardware software guard extensions (SGX) for the back-end to scrape HTTPS-enabled websites and serve source-authenticated data to rely on smart contracts. It executes its main role as a relevant piece of code in an SGX enclave, which protects against malevolent processes. It can attest (prove) to a remote client that the client is interacting with a legitimate, SGX-backed instance of the TC code. Moreover, with highly confidentiality, it enables private data requests with encrypted parameters. Smart-contract logic within TC can execute if the system permits secure use of user credentials to scrape access-controlled online data sources.

Conclusion of the section on scanning tools which is we mention above and their vulnerability detecting results on smart contract presented in Table 5 with comparative aspects.

6 | DISCUSSION AND FUTURE RESEARCH DIRECTIONS

Several studies have examined the issues related to BT systems and their vulnerabilities. The major solution to secure BT systems is that all nodes must upgrade their versions in order to solve a particular security issue. However, this has made the security mechanism challenging to push all node upgrades in the blockchain network because this is a radical change to the cryptocurrency protocol that introduces a new rule into the blockchain network called hard fork. Contrarily, a hard fork is a permanent divergence from the previous version of the blockchain, and thus, nodes running previous versions will no longer be accepted by the newest version. This has become the first major limitation of BT because there will always be a misunderstanding among network participants about the blocks or transactions initiated through a hard fork. The second major limitation of BT systems is their privacy. In the case of public BT, transactions may appear private because they are not directly tied to a user identity. However, they are recorded in a public ledger. Thus, transaction patterns can be observed, and it is possible to link a user identity to an address. A major contribution of BT is the degree of transparency and decentralization that it provides along with an adequate level of security and privacy that was previously deemed impossible. However, no solution for transaction privacy is perfect. Real attacks on Ethereum and Bitcoin systems in an anonymous environment make it difficult to identify SPAM or DoS transactions even over a significant time, such as a month. Thus, it is crucial to manage financial and network risks on BT-based cryptocurrency networks because there is lack of methods that prevent security threats, such as 51% attacks.

Potential security threats are still issues in BT system development that need to be overcome. Consistent efforts are being made by researchers and developers to address issues related to BT systems security. These efforts are mainly focusing on combining various cutting-edge technologies, such as machine learning, artificial neural networks, and deep learning into blockchain security mechanisms.

Generally, blockchain is difficult to manipulate but it is still possible with the DAO, Mt. Gox, and Bitfinex attacks. Recently, machine learning (ML) techniques are being used to predict malicious and normal traffic where unsupervised learning is widely used for anomaly and novelty detection. With such techniques, a large data that is reduced and set into a smaller number of common labels can be understood for a particular transaction or account to be

"normal." With a trained definition, transactions or accounts can determine the extent to which they are anomalous by comparing them with the global average or to a recent historical average. These anomaly detection systems can then be used to alert users about unusual events on the blockchain or within a subset of accounts or transactions. With supervised technique on a blockchain, the authenticity of an account can be predicted and classified if it is a fraud or spam attack. A protocol-level data can also be used that are available on chains, such as transaction data, and extract features of accounts to train ML algorithms for the above purposes. Models that provide actionable insights about new accounts and recent behavioral data must satisfy certain requirements. For example, it is required that they are updated in real time and that the features being used for classification and prediction are reliable and complete when a model is run. This means that features that can be used to classify "old" accounts, such as "whether a contract eventually self-destructed" (SUICIDE) cannot be applied to accounts in real time. Because the value of a feature may change over time and its true value is not really known when a model runs. In fact, blockchain is ideal for storing highly sensitive, personal data because it holds data in an encrypted state. It means that account holders must keep their private keys safe. Because RSA algorithms and the elliptic curve cryptography standard which are the most widely used public key algorithms can all be broken by quantum computers. Information on a blockchain database is stored indefinitely in every full node in the network because the database is append-only and immutable. Therefore, data storage imposes a huge cost on a decentralized network, where every full node must store increasing amounts of data. As a result, storage remains a significant hurdle for blockchain applications. For example, a smart contract can erase its code using SUICIDE or SELFDESTRUCT, but the address of the contract will not be erased. In addition, many smart contracts contain no code or contain the same code as Ethereum, while many others contain code that will never be executed. The time taken to process a chain increases over time, making this a chaotic data environment. AI technology may introduce new methods for optimizing data sharding. This can help blockchains to make smarter decisions about data storage and maintenance as well as provide reliable data sources. Blockchain cryptocurrency implementations use centralized pools for mining. Each pool is comprised of miners with different power who all try to verify new transaction blocks. Thus, many miners are wasting energy. In the future, AI may improve mining pools, optimizing this process and decreasing the cost of mining.

7 | CONCLUSIONS

PoW is the most popular consensus mechanism and an underpinning technology that maintains Bitcoin. It is clear that the construction of the Bitcoin with PoW and a secure timestamping service provides a strong security solution. But it seems that this solution is subjected to a number of security threats, for example, double-spending (or race attacks) attack. Some blockchain platforms were designed to be permission less by adopting PoW and improved privacy and anonymity, for example, LiteCoin that uses Segwit and allows technologies like Lightning Network. ZeroCoin is a cryptographic extension to Bitcoin that provides unlikable and untraceable transactions by using zero-knowledge proofs. This design has good scalability, low transaction mining time, anonymous, and cheaper in terms of nodes participating in the network though hard to remark on its survivability. The scalability of the network, the continuously decreasing rewards, increasing transaction fee, and the security and privacy threats are the persistent issues, which need to be addressed for a secure and successful system. Ethereum developed PoS consensus mechanism that is significantly faster and efficient than PoW system, since technically anyone could become a miner, and it offers a linear scale relative to the percentage of blocks which a miner could confirm because it is based on the cryptocurrency quota owned mechanism. In terms of security, PoS mechanism also has its own drawbacks, such as nothing-at-stake. Although, BitShares developed new consensus model (DPoS) which is an advanced variant of PoS that provides a high level of scalability. DPoS is designed as an implementation of technology-based democracy using voting and election process to protect blockchain from malicious usage. In aspects of security, DPoS systems are vulnerable to centralization as a number of witnesses are strictly limited.

In fact, BT has radically changed the transaction-based industries. However, there are some security concerns and risks that prevent this technology from being used as a general platform in other implementations around the world. Several studies and practical implementations provide solutions against these risks. However, robust and effective security solutions that can ensure proper functioning of BT in the future are still remaining as challenges and open research issues. With the rapidity of its growth and development, it is believed that BT may soon become a very common technology in business areas as well as industrial sectors.

ACKNOWLEDGEMENTS

This work was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea Government (MSIT) (No. 2018-0-00539-001) and Development of Blockchain Transaction Monitoring and Analysis Technology.

CONTRIBUTION

This study presents a comprehensive review on the security and privacy aspects of blockchain—a distributed ledger technology. The study explores emerging attack vectors in which various user security and vulnerabilities to blockchain are identified. In fact, blockchain technology is being applied to various fields that range from finance, to gaming, gambling, supply chain, manufacturing, trade, and ecommerce. Therefore, we discuss the capability of various state-of-the-art security solutions and intrusion detection methods that are proposed over the years addressing the existing security and privacy challenges in crypto services. In particular, this study focuses on security challenges and possibilities of attacks on current cryptocurrencies wherein the issues of user privacy and transaction anonymity are primarily underlined. Besides, this study provides a comprehensive technical review on decentralized digital currencies emphasizing Bitcoins and Ethereum.

Previous studies have explored the technical architecture of Bitcoin and blockchain technologies. These studies have discussed the fundamental background knowledge of crypto currency, preliminary overview of its use and functionalities, and its privacy. With regards to the prevalence of crypto currencies with current security challenges, these studies have done little on the vulnerabilities and privacy to blockchain. In view of the dynamic development attack methods by the attackers, the existing approaches these studies discussed are slightly outdated. In the current study, we did not attempt to solve any new challenge and threats to the current security situation. Nevertheless, we present an overview of the blockchain security, and highlight vulnerabilities and threats along with its available countermeasures that may be useful for security experts. The main contributions of this study are presented below.

Firstly, this study presents the critical background knowledge required for blockchain technology and highlights its functionalities. The purpose is to provide the primary knowledge of distributed ledger technology, such as proof-of-work, and smart contract. This may be required to understand the working methodology, benefits, and challenges that are associated with the use of this technology. Second, this study demonstrates the existing security, privacy and detection methods (algorithms) that are used for blockchain technology. At various levels, we investigate the possibilities and types of potential attacks, which may include both practical and theoretical risks of technology. Furthermore, we discuss the limitations of the state of-the-art solutions that address security threats and enable strong privacy. Finally, based upon our review, this study provides directions for further study to tackle vulnerabilities to blockchain technology.

The major benefit of blockchain may mitigate the risk that a cyberattack directed to a single point brings down the entire network. However, a coded intrusion or any single system vulnerability could have wider negative consequences on the state of a system. For example, if any attackers were to break the system might have access not only to the information stored at the point of attack but also to the full breadth of information recorded on the ledgers. As such, the security and privacy issues of blockchain technology become critical in cyber security. Due to this reason, it is crucial to understand the scope and impact of security and privacy challenges in blockchain to predict the possible damage caused by these threats and to corroborate whether the current technology is enough to withstand constant hacking.

ORCID

Huru Hasanova https://orcid.org/0000-0003-4316-2395
Myung-Sup Kim https://orcid.org/0000-0002-3809-2057

REFERENCES

1. Yli-Huumo J, Ko D, Choi S, Park S, Smolander K. Where is current research on Blockchain technology?—a systematic review. *PLoS ONE*. 2016;11(10):e0163477. https://doi.org/10.1371/journal.pone.0163477

2. Underwood S. Blockchain beyond bitcoin. *Communications of the ACM*. 2016;59(11):15-17.

3. Brito J, Shadab H, Castillo A. Bitcoin financial regulation: securities, derivatives, prediction markets, and gambling. *Colum Sci & Tech L Rev*. Apr 2014;16:144.
4. Fanning K, Centers DP. Blockchain and its coming impact on financial services. *J Corp Acc Financ*. Jun 2016;27(5):53-57.
5. Eyal I., Gencer A. E., Sirer E. G., and Van Renesse R. bitcoin-ng: a scalable blockchain protocol. In NSDI, 2016:45–59.
6. Antonopoulos AM. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media, Inc; 2014.
7. Bradbury D. The problem with bitcoin. *Comput Fraud Secur*. 2013;2013(11):5-8.
8. Singh S, Singh N. Blockchain: future of financial and cyber security. In: *Contemporary Computing and Informatics (IC3I)*, 2016 2nd International Conference on. IEEE; Dec 2016:463-467.
9. Nakamoto S. Bitcoin: a peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf, retrieved on 28/04/2018
10. Crosby M, Pattanayak P, Verma S, Kalyanaraman V. Blockchain technology: beyond bitcoin. *Appl Innov*. 2016;2:6-10.
11. Swan M. *Blockchain: Blueprint for a New Economy*. O'Reilly Media, Inc; 2015.
12. Zheng Z, Xie S, Dai H, Chen X, Wang H. An overview of blockchain technology: architecture, consensus, and future trends. In: *Big data (BigData congress)*, 2017 IEEE international congress on. IEEE; 2017:557-564.
13. Sinha SR, Park Y. Dealing with security, privacy, access control, and compliance. In: *Building an Effective IoT Ecosystem for Your Business*. Cham: Springer; 2017:155-176.
14. Tschorsch F, Scheuermann B. Bitcoin and beyond: a technical survey on decentralized digital currencies. *IEEE Commun Surv Tutorials*. 2016;18: .
15. Conti M, Lal C, Ruj S. A survey on security and privacy issues of bitcoin. *IEEE Commun Surv Tutorials* https://doi.org/10.1109/COMST.2018.2842460. 20(4):3416-3452.
16. Atzei N, Bartoletti M, Cimoli T. A survey of attacks on ethereum smart contracts (sok). In: *International Conference on Principles of Security and Trust*. Springer; 2017:164-186.
17. King S, Nadal S. *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake*. self-published paper; 2012.
18. Baliga A. Understanding blockchain consensus models. Tech. Rep., persistent systems ltd. *Tech Rep*. 2017 .
19. [Online]. Ethereum on Github. Available: https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ
20. Proof of Stake versus Proof of Work: White Paper https://bitfury.com/content/downloads/pos-vs-pow-1.0.2.pdf
21. Vasin P. Blackcoin's proof-of-stake protocol v2. URL: https://blackcoin.co/blackcoin-pos-protocol-v2-whitepaper.pdf. 2014
22. Xu JJ. Are blockchains immune to all malicious attacks? *Financ Innov*. Dec 2016;2(1):25.
23. Karame G, Androulaki E, Capkun S. Two bitcoins at the price of one? Double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*. 2012;248:2012.
24. Ren L. *Proof of Stake Velocity: Building the Social Currency of the Digital Age*. Self-published white paper; 2014.
25. Extance A. The future of cryptocurrencies: Bitcoin and beyond. *Nature News*. 2015;526(7571):21.
26. Scaife N, Carter H, Traynor P, Butler KR. Cryptolock (and drop it): stopping ransomware attacks on user data. In: *Distributed Computing Systems (ICDCS), 2016*. IEEE 36th International Conference on. IEEE; 2016:303-312.
27. Johnson D, Menezes A, Vanstone S. The elliptic curve digital signature algorithm (ECDSA). *Int J Inf Secur*. 2001;1(1):36-63.
28. Van Dam W, Shparlinski IE. Classical and quantum algorithms for exponential congruence's. In: *Workshop on Quantum Computation, Communication, and Cryptography*. Berlin, Heidelberg: Springer; 2008:1-10.
29. Vedral V, Morikoshi F. Schrödinger's cat meets Einstein's twins: a superposition of different clock times. *Int J Theor Phys*. 2008;47(8):2126-2129.
30. Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. In: *Foundations of Computer Science, 1994 Proceedings*., 35th Annual Symposium on. Ieee; 1994:124-134.
31. Lakshmanan T, Madheswaran M. Security and robustness enhancement of existing Hash algorithm. In: *2009 International Conference on Signal Processing Systems*. IEEE; 2009:253-257.
32. Kiayias A., Konstantinou I., Russell A., David B., and Oliynykov R. A provably secure proof-of-stake blockchain protocol. IACR Cryptology ePrint Archive, 2016
33. Eyal I, Sirer EG. Majority is not enough: Bitcoin mining is vulnerable. In: *International conference on financial cryptography and data security*. Berlin, Heidelberg: Springer; 2014:436-454.
34. Bonneau J. Why buy when you can rent? In: *International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer; 2016:19-26.
35. Feder A, Gandal N, Hamrick JT, Moore T. The impact of DDoS and other security shocks on bitcoin currency exchanges: evidence from Mt. *Gox J Cybersecurity*. 2018;3(2):137-144.

36. Vasek M, Thornton M, Moore T. Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In: *International conference on financial cryptography and data security 2014 Mar 3*. Berlin, Heidelberg: Springer; :57-71.
37. Johnson B, Laszka A, Grossklags J, Vasek M, Moore T. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In: *In International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer; 2014:72-86.
38. Douceur JR. The sybil attack. In: *International workshop on peer-to-peer systems*. Berlin, Heidelberg: Springer; 2014:251-260.
39. Heilman E, Kendler A, Zohar A, Goldberg S. Eclipse attacks on Bitcoin's peer-to-peer network. In: *USENIX Security Symposium*; 2015:129-144.
40. Kermarrec AM, Van Steen M. Gossiping in distributed systems. *ACM SIGOPS Operating Systems Review*. 2007;41(5):2-7.
41. Marcus Y., Heilman E., and Goldberg S. Low-resource eclipse attacks on Ethereum's peer-to-peer network. Report 2018/236, 2018.
42. Stock B., Göbel J., Engelberth M., Freiling F. C., and Holz T. Walowdac-analysis of a peer-to-peer botnet. In *Computer Network Defense (EC2ND)*, 2009 European conference on IEEE; 2009:13–20.
43. Andrychowicz M, Dziembowski S, Malinowski D, Mazurek Ł. On the malleability of bitcoin transactions. In: *International Conference on Financial Cryptography and Data Security*. Berlin, Heidelberg: Springer; 2015:1-18.
44. Empty Accounts and the Ethereum State https://www.ethnews.com/vitalik-buterin-on-empty-accounts-and-the-ethereum-state retrieved on 24/07/2018
45. [Online]. Delegated Proof-of-Stake Consensus http://docs.bitshares.org/bitshares/dpos.html retrieved 10.02.2018
46. [Online]. Notes on Blockchain. Governance https://vitalik.ca/general/2017/12/17/voting.html retrieved 10.02.2018
47. EOS.IO Technical White Paper v2, Available on: https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md retrieved 10.02.2018
48. [Online]. Dan Larimer's DPOS Consensus Algorithm - The Missing White Paper https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper retrieved 10.02.2018
49. [Online]. Governance, Part 2: Plutocracy Is Still Bad https://vitalik.ca/general/2018/03/28/plutocracy.html retrieved 10.02.2018
50. [Online]. Attacks on the network https://forums.eosgo.io/discussion/71/attacks-on-the-network retrieved 10.02.2018
51. Myles S., Kyle S., and Tushar J., Delegated proof of stake: features & tradeoffs. 2018.
52. [Online]. Response to Cosmos white paper's claims on DPOS security. https://steemit.com/steem/@dantheman/response-to-cosmos-white-paper-s-claims-on-dpos-security retrieved 10.02.2018
53. Chen T, Li X, Luo X, Zhang X. Under-optimized smart contracts devour your money. In: *Software Analysis, Evolution and Reengineering (SANER)*, 2017 IEEE 24th International Conference on. IEEE; Feb 2017:442-446.
54. Luu L., Chu D. H., Olickel H., Saxena P., and Hobor A. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. ACM; 2016:254–269.
55. [Online]. Thinking smart contract security. Available: https://blog.ethereum.org/2016/06/19/thinking-smart-contract-security retrieved on 15/07/2018
56. Rodrigues, Usha, Law and the Blockchain. Iowa Law Review, Vol. 104, 2018, Forthcoming; University of Georgia School of Law Legal Studies Research Paper No. 2018-07. Feb 2018. Available at SSRN: https://ssrn.com/abstract=3127782
57. [Online]. CRITICAL UPDATE Re: DAO Vulnerability https://blog.ethereum.org/2016/06/17/critical-update-re-dao-vulnerability/: Retrieved on 06/07/2018.
58. [Online]. Available: https://paritytech.io/blog/
59. [Online]. Available: https://github.com/paritytech/parity/blob/4d08e7b0aec46443bf26547b17d10cb302672835/js/src/contracts/snippets/enhanced-wallet.sol#L424
60. [Online]. Available: https://blog.zeppelin.solutions/on-the-parity-wallet-multisig-hack-405a8c12e8f7
61. [Online]. Available: https://gist.github.com/ethanbennett/7396bf3f61dd985d3426f2ee184d8822#parity-multisig-wallet
62. [Online]. Quorum whitepaper https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf
63. [Online]. Corda security model. https://docs.corda.net/releases/release-M10.1/key-concepts-security-model.html
64. [Online]. Hyperledger. Available: https://www.hyperledger.org/ retrieved on 06/07/2018
65. [Online]. Hyperledger Fabric. Available: https://hyperledger-fabric.readthedocs.io/en/release-1.1/ retrieved on 06/07/2018
66. Tien D., Ji W., Gang Ch., Rui L., Beng Ch., Kian-Lee T., BLOCKBENCH: A Framework for Analyzing Private Blockchains. arXiv: 1703.04057v1 [cs. DB]. 2017.
67. Elli A., Artem B., Vita B., Christian C., Konstantinos Ch., Angelo D., David E., Christopher F., Gennady L., Yacov M., Srinivasan M., Chet M., and Binh N., Hyperledger fabric: a distributed operating system for permissioned blockchains. EuroSys'18. Apr 2018, Porto, Portugal
68. [Online]. Available: https://fabric-sdk-node.github.io/
69. [Online]. Available: https://nodejs.org/en/blog/vulnerability/oct-2017-dos/

70. [Online]. Available: https://nodejs.org/en/blog/vulnerability/march-2018-security-releases/
71. Graham Sh. Security, Hyperledger Fabric version: 1.1 Assessment Technical Report. 2017.
72. [Online]. Docker security. Available: https://docs.docker.com/engine/security/security/#kernel-namespaces retrieved on 12/07/2018
73. Mouat A. UsingDocker. Developing and Deploying Software with Containers. 2015.
74. [Online]. Stuart P., Confidentiality in Private Blockchain., http://kadena.io/docs/Kadena-ConfidentialityWhitepaper-Aug2016.pdf
75. [Online]. Membership Service Providers (MSP) https://hyperledger-fabric.readthedocs.io/en/release-1.2/msp.html
76. [Online]. Rust programming language https://www.rust-lang.org/en-US/
77. Punitive-proof-of-stake-algorithm https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/ retrieved on 08/07/2018
78. Vitalik B. and Virgil G. Casper the Friendly Finality Gadget, white paper, arXiv:1710.09437v2 [cs.CR]. Nov 2017
79. Jae K. Tendermint: Consensus without Mining. URL. https://tendermint.com/static/docs/tendermint.pdf retrieved on 08/07/2018
80. Loi L., Yaron V., Jason T., and Prateek S., SMARTPOOL: Practical Decentralized Pooled Mining, SEC'17 Proceedings of the 26th USENIX Conference on Security Symposium, 2017:1409–1426.
81. [Online]. Available: https://en. Bitcoin.it/wiki/Protocol_documentation#BlockTransactions
82. Muhammad S., My T., Aziz M. POSTER: deterring DDoS attacks on blockchain based cryptocurrencies through mempool optimization. Proceeding ASIACCS '18 Proceedings of the 2018 pp 809–811. https://doi.org/10.1145/3196494.3201584. 2018
83. Ting Ch., Xiaoqi L., Ying W., Jiachi C., Zihao L., Xiapu L., Man H., and Xiaosong Z. An Adaptive Gas Cost Mechanism for Ethereum to Defend Against Under-Priced DoS Attacks.13th International Conference, ISPEC 2017
84. Petar T., Andrei D., Drachsler C., Arthur G., Florian B. Securify: Practical Security Analysis of Smart Contracts. arXiv:1806.01143v1 [cs.CR]. 2018
85. Sergei T., Ekaterina V., Ivan I., Ramil T., Evgeny M., Yaroslav A. SmartCheck: Static Analysis of Ethereum Smart Contracts. 2017
86. Ahmed K, Andrew M, Elaine S, Zikai W, Charalampos P. Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. *Proc. IEEE Symp. Secur. Privacy (SP)*. 2016; :839-858.
87. Fan Z, Ethan C, Kyle C, Ari J, Elaine S. Town crier: An authenticated data feed for smart contracts. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16*. ACM; 2016:270-282.
88. [Online]. Vector76. The vector76 attack. Available onhttps://bitcointalk.org/index.php?topic=36788.msg463391#msg463391, retrieved on 28/04/2018.
89. [Online]. Solidity. Available: http://solidity.readthedocs.io/en/develop/, retrieved on 02/08/2018
90. [Online]. Known Attacks. Available: https://consensys.github.io/smart-contract-best-practices/known_attacks/ retrieved on 02/08/2018
91. The Finney Attack, Available from https://bitcoincoreacademy.com/the-finney-attack, retrieved on 28/04/2018

AUTHOR BIOGRAPHIES

Huru Hasanova received her BS and MS degrees in Computer Science from Azerbaijan Technical University in 2008 and 2010, respectively. Now, she is a PhD student in The Korea University at Department of Computer and Information Science. Her research focuses on internet security and Distributive Ledger Technologies

Ui-Jun Baek received his BS Degree in Computer Science and Engineering from Korea University, Korea in 2018. He joined Korea University, Korea, in March 2018, where he is studying for master's degree in the Department of Computer and Information Science. His research interests include traffic classification, traffic analysis, and automatic signature generation.

Mu-Gon Shin is studying at Korea University for bachelor's degree. He is currently working as an undergraduate researcher in the Department of Computer and Information Science. His research interests include traffic classification, traffic analysis, and automatic signature generation.

Kyunghee Cho received her BA in English Literature from Syungkyul University in 1994, her MA in English Education from Yonsei University in 1997, and her PhD in Computer Science from Korea University in 2018. Her research interests include service and network management, and internet security, home network system related to internet of things.

Myung-Sup Kim received his BS, MS, and PhD degrees in Computer Science and Engineering from POSTECH, Korea, in 1998, 2000, and 2004, respectively. From September 2004 to August 2006, he was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University, Korea, in September 2006, where he is currently working as an associate professor in the Department of Computer and Information Science. His research interests include Internet traffic monitoring and analysis, service and network management, and Internet security.

How to cite this article: Hasanova H, Baek U, Shin M, Cho K, Kim M-S. A survey on blockchain cybersecurity vulnerabilities and possible countermeasures. *Int J Network Mgmt*. 2019;e2060. https://doi.org/10.1002/nem.2060

APPENDIX A

POW AND POS-BASED BT VULNERABILITIES

APPENDIX TABLE 1 Major attacks on BT system using POW and POS

	Attack	Description	Adverse Effects	Weaknesses of Technology
Consensus-level attack	Double spending[22]	User makes more than one payment using one form of funds	Vendor may lose their product	Fundamental problem of blockchain protocol
	Finney attack[91]	Form of double spending where is attacker secretly premined block and broadcast it for purpose of double spending	Merchant lose some goods or service	POW-based consensus protocol
	Vector76 attack[88]	Type of one-confirmation attack that attacker premines and not publish a block, which includes a deposit transaction to the target (eg, an exchange or similar service with pooled hot wallet in which all funds are held in a big pool, not individual addresses for each user) in this block	Attacker has received coin from withdraw service and transferred between two addresses both are under the attacker control	POW-based consensus protocol
	Brute force attack[15]	Brute-force attack is an advance form of the Finney.	Has similar effect as Finney attack and it is a successful double spending	POW-based consensus protocol
	Majority attack 51%	Control over 50% hashpower from total hashpower	The >50% attack is worst-case scenario in the blockchain network. Attacker can reverse transactions and perform double-spending attack Can modify the ordering of transactions and control the confirmation operation of normal transactions.	Fundamental problem of blockchain system. POW base system more vulnerable. POS-based blockchain also has several form of 51% attack such as *finality reversion*, *attack is liveness denial*, *censorship attack*
	Nothing-at-stake[19]	When a stake moves, the existing majority of stakeholders remain honest wherein the past account keys can be negotiated that have no stake at the present. Thus, this can be a major weakness in PoS.	Set of malicious shareholders from the past can form a different blockchain using old accounts	Nothing-at-stake important flaw of proof-of-stake algorithms

(Continues)

APPENDIX TABLE 1 (Continued)

	Attack	Description	Adverse Effects	Weaknesses of Technology
Mining pool attack	Selfish mining[16]	Attacker not broadcasting blocks immediately on the network and publish block on selective time.	Potentially allowing for 51% attacks	PoW-based blockchain is more vulnerable.
	Time base attack Long-range attack[16]	POW-based system: Attacker instead of starting a forking 1 or more blocks back, start forking short after first block even at the genesis block. PoS-based system: Attacker with 1% of all coins start make own chain short after genesis block. If attacker success may produce many block and will lead on long chain.	It has similar effect as traditional 51% attack	Long-range attack is POW-based system vulnerability and naively implemented PoS-based system also
	Bribery attacks[34]	Adversary bribe miners to mine	Increasing chance to double spending	Both POW and POS-based system can happen. But POS more vulnerable to that because POS bribe attack cost lower than POW bribe attack.
Network-level attack	Transaction malleability[42,43]	Attacker can change unique signature of a Bitcoin transaction (TXID) before a transaction is put into a block other words it is happen only before transaction confirmed on the Bitcoin network.	Exchange system will lose funds because of double spending.	Implementation vulnerabilities on Bitcoin protocol which assist the DoS attacks
	Wallet theft[26]	Stole or destroy private key of users. Attack may happen not proper implementation of cryptographic solution for security.	Wallet can be lost not access to account	It is fundamental problem
	DDoS[36,37]	Is aimed to disrupt the normal operation of the cryptocurrency network by flooding the nodes	Honest node will be isolated from network due to massive spam request	POW more vulnerable
	Sybil[38]	Sybil attack occurs when the networks without admission controls, it is can allow to user to create many id and monopolize the consensus process.	With dominance allows to the attacker or attacker groups may change protocol confirmation rules	POW more vulnerable
	Eclipse or netsplit[38,39]	Victim node essentially isolated from the rest of the network and its view can be manipulated by the attacker. Targets of attack nodes that accept incoming connections because not all nodes accept incoming connections	Attacker can launch a 51% attack with 40% mining power	POW more vulnerable
	Time jacking[15]	Attacker speed-up the majority of miner's clock	Isolate a miner and consume of its resources, affected the mining difficulty calculation process	Both POW and POS-based system can happen

(Continues)

APPENDIX TABLE 1 (Continued)

Attack	Description	Adverse Effects	Weaknesses of Technology
Tampering[15]	Attacker can exploit these measures in order to effectively delay the propagation of transactions and blocks to specific nodes-without causing a network partitioning in the system.	Adversary to easily mount denial-of-service attacks, considerably increase its mining advantage in the network, and double-spend transactions	POW more vulnerable

APPENDIX B

SMART CONTRACT–BASED BT VULNERABILITIES

APPENDIX TABLE 2 Common pitfalls and vulnerabilities in smart contracts

Pitfalls	Description	Adverse Effects	Recommendations
"Bockhash" usage[89]	Blockhash usage is similar with block timestamp not recommended that to use on crucial components, both of them can be influenced by miners. Not using proper Blockhash usage easily can manipulate of output.	In the mining community attacker may run a casino payout function on a chosen hash and just retry a different hash if they did not receive any money.	The current block timestamp must be strictly larger than the timestamp of the last block[89] Use function *getChances()* for check[89]
Balance equality[84]	It occur when not use proper code for checking for strict balance equality	Attacker may manipulate contract logic by forcibly sending ether to a contract without triggering fallback function.	Use nonstrict inequality on balances[84]
Call to the unknown/DoS with unexpected revert[16]	Callee/recipient function to invoke and to transfer ether may have the side effect or function is not exist	Primitively written fallback function may be related with other vulnerabilities such as "type cast," "unpredictable state," also causes a permanent denial of service to the contract	Recommended to possibly avoid external calls
DoS by external Contract[84]	When conditional statement (if, for, while) depend on an external call: The callee may permanently fail (throw or revert), preventing the caller from completing the execution.	Malicious bidder can become the leader while making sure that any refunds to their address will always fail. In this way, they can prevent anyone else from calling the *bid()* function, and stay the leader forever.[71]	Conditional statement should not depend on an external call. Set up a pull payment system[90]
Ether lost in transfer[16]	If ether is sent to an "orphan" address that does not actually belong to any private key or contract that ether will be lost and cannot be retrieved.	Ether will lost permanently if sent to orphan address. There are no way to find out an address is orphan or not.	Use the withdraw pattern[89]
Exception disorders or mishandled exceptions[16]	The exception disorder vulnerability happen due to Solidity is inconsistent on exception handling and it is depended on the way contracts call each other. Due to different types of exceptions calling process can be fail. If A call B and if B runs abnormal, B return false and stop running. Then A must check the return value explicitly for verify that call has been executed properly. In case A not checks the exception information it may related to vulnerable.	The effect is that the calling contract transaction is entirely reverted and all gas is lost.	Use symbolic execution tools: Oyente for detect bug[54]

(Continues)

APPENDIX TABLE 2 (Continued)

Pitfalls	Description	Adverse Effects	Recommendations
Gasless send[16]	If the *callee* of a send is a contract with a relatively expensive fallback function, then the amount of gas the caller is limited to 2300 units for sending ether to an address will is insufficient, and an out-of-gas exception will be thrown.	In Ethereum for each transaction need specific gas value. If there is not enough gas to provided call function then transaction will be fail. Gasless send bug makes over gas consumption	Use a.*transfer()*[84] Consider to develop function that no need too much gas
Gas costly pattern and Loops[53]	There is not having a fixed number of iterations on loops. Ethereum set *gasvalue* for each transaction. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.	Expensive computation inside loops may exceed the block gas limit and can cause gas consumption or ether lost	Avoid loops with big or unknown number of steps[84] Use selfdestruct(x)[89] Use "withdraw" pattern[89] Reduce expensive operations
Generating randomness[16]	For making unpredictable of the content of future blocks using random numbers is secure way. However, which transactions are put in a block and in which order is under miner control. Dishonest miner may try to craft own block so to bias the outcome of the pseudo-random generator	A malicious miner could arrange their block to influence the outcome of this random number generation.	Timed commitment protocols[16]
Immutable bugs/ mistakes[16]	Immutable bugs which refer to a bug or any programming code pitfalls that, which is cannot be altered after publish smart contract.	Can cause in various attacks	Make bugs impossible Localize bugs
Lack of transactional privacy[66]	The present form of blockchain and smart contracts technologies mainly vulnerable to leakage of transactional privacy. Because all public keys are visible to everyone in the network.	Transaction patterns can be observed and it is possible to link your identity to the address	Use mixing and anonymous solution to achieve anonymity. Use of "hawk" tool[66] Secret contracts (Enigma's secret contracts) Use of "town crier (TC)" tool[67]
Malicious libraries[84]	Malicious code on library may collect sensitive information on infected smart contract	Third-party or external libraries can be malicious	Avoid external dependencies or ensure that third-party library source is authorized[84]
Redundant fallback function	This function is no name and cannot have arguments, also not return anything and should be external visibility. Furthermore, when contract receives plain ether this function is executed. The fallback function has to marked payable for received ether. If not the contract cannot receive ether with regular transactions. Function can only rely on 2300 gas being available.	Some fallback function requires more than 2300 gas. Such as, writing to storage, creating a contract, sending ether. DAO attack caused wrong design of fallback function	Avoid creating big fallback functions Instructing users to explicitly call the function with more gas
Reentrancy[16]	In some cases, a contract's fallback function allows it to re-enter a caller function before it has terminated. This can result in the repeated execution of functions intended only to be executed once per transaction	Bug refers that functions that could be called repeatedly and if user's balance is not set to 0 until the end of function. Then other invocations will succeed, and will withdraw the balance many times over in a single transaction.	Use *send()* instead of *call.value()*[89] Use Oyente tool for detect bug[54]

(Continues)

APPENDIX TABLE 2 (Continued)

Pitfalls	Description	Adverse Effects	Recommendations
			Do not call an external function until the internal work terminate
Send instead of transfer[89]	Send is the low-level counterpart of transfer. If the execution fails, the current contract will not stop with an exception, but send will return false[89]	In order to it can cause lost ether or gas consumption	Check the return value of *send*[89]
Timestamp dependence[86]	Blockchain technology use timestamp for each block. However, trigger conditions of some smart contract depend on timestamp, which is set by the miner according to its local system time.	The timestamp of the block can be manipulated by the miner.	Do not use for critical components of the contract Use block numbers and average time between blocks to estimate the current time. Use secure sources of randomness, such as RANDAO[84]
Transaction-ordering dependence[86]	This problem occurs when two dependent transactions that invoke the same contract are included in one block. The order of executing transactions control by the miner. But if those transitions were not executed in the right order an adversary can successfully launch an attack.	This can be troublesome for things like decentralized markets, where a transaction to buy some tokens can be seen, and a market order implemented before the other transaction gets included.	Use a precommit scheme[89] Use Oyente tool for detect bug[54]
Unchecked external call[16]/ untrustworthy data feeds (oracles)	External contracts are that they can take over the control flow, and make changes to data that the calling function was not expecting.	This class of bug can take many forms such as reentrancy, and both of the major bugs that led to the DAO's collapse were bugs of this sort.	Check the return value of *send*[89] Secret contracts (Enigma's secret contracts) Use caution when making external calls Use of "town crier (TC)" tool[87]
Unchecked math[84]	Smart contracts mainly control upon arithmetic operations, such as reiterating over an array or computing balance amounts to send to a participant. An overflow occurs when a number gets incremented above its maximum value. However, Solidity can handle up to 256-bit numbers (up to 2^{256}-1), so incrementing by 1 would result into 0. When the number is unsigned, decrementing will underflow the number, resulting in the maximum possible value and it is arithmetic vulnerability.	Especially in the underflow case, token holder has X tokens but attempts to spend X + 1. If the code does not check for it, the attacker might end up being allowed to spend more tokens than it had and have a maxed-out balance	Use SafeMath library[84] Use Smartcheck tool[84]
Using *tx.origin*[84]	Use of *tx.origin* for detecting the contract caller can make the contract vulnerable. If there is a chain of calls, *msg.sender* points to the caller of the last function in the call chain. Solidity's *tx.origin* attribute allows a contract to check the address	Using tx.Origin makes you vulnerable to attacks such as, phishing or cross-site scripting	Avoid to use *tx.origin* for authorization[89] Use *msg.sender* for authorization[89]

(Continues)

APPENDIX TABLE 2 (Continued)

Pitfalls	Description	Adverse Effects	Recommendations
	that originally initiated the call chain, and not just the last function call		
Visibility or exposed functions (fallout)[16]	Everything used in a smart contract is publicly visible to all external observers, even local variables and state variables marked private.	Contract fields marked private are not guaranteed to remain secret to set a private field.	Avoid ambiguity: Explicitly declare visibility level[84]