IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 20, 2019, accepted February 18, 2019, date of publication February 26, 2019, date of current version March 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2901858

A Survey on Long-Range Attacks for Proof of Stake Protocols

EVANGELOS DEIRMENTZOGLOU[1], GEORGIOS PAPAKYRIAKOPOULOS[2], AND CONSTANTINOS PATSAKIS[1], (Member, IEEE)

[1]Department of Informatics, University of Piraeus, 18534 Piraeus, Greece
[2]Cisco Security Services, 151 25 Marousi, Greece

Corresponding author: Constantinos Patsakis (kpatsak@unipi.gr)

This work was supported in part by the University of Piraeus Research Center.

ABSTRACT Despite common arguments about the prevalence of blockchain technology, in terms of security, privacy, and immutability, in reality, several attacks can be launched against them. This paper provides a systematic literature review on long-range attacks for proof of stake protocols. If successful, these attacks may take over the main chain and partially, or even completely, rewrite the history of transactions that are stored in the blockchain. To this end, we describe how proof of stake protocols work, their fundamental properties, their drawbacks, and their attack surface. After presenting long-range attacks, we discuss possible countermeasures and their applicability.

INDEX TERMS Blockchain proof of stake long-range attacks.

I. INTRODUCTION

Around a decade ago Satoshi Nakamoto introduced Bitcoin [1]. Despite the revolutionary nature of Bitcoin that made it famous around the world, there are far more potentials from the underlying technology. To solve the problem of maintaining the order of transactions and to avoid double-spending of bitcoins, an immutable public ledger was needed, a place that everyone could see all the transactions that have been performed and that cannot be altered. To this end, Nakamoto used strong cryptographic primitives to introduce the blockchain technology, a distributed peer-to-peer linked-structure.

This structure groups together transactions into *blocks* which are validated by groups of blockchain users. In traditional blockchains, e.g., Bitcoin, users compete with each other in solving difficult cryptographic/mathematical problems which are easy to verify. This process is called ''mining,'' and the winner gets new coins as a reward for her services. These blockchains are therefore based on the concept of *Proof of Work (PoW)*. Practically, we consider the user to be trustworthy because she spent a considerable amount of computational effort to verify some transactions. On the contrary, in *Proof of Stake (PoS)* protocols, the users that validate transactions are chosen based on their wealth (*stake*). Therefore, the coins are generated in the initialisation of the blockchain and to motivate validators; they get as a reward a share of each transaction they validate (transaction fees). In this regard, users are considered to be trustworthy since they ''stake'' a part of their property in block validation.

PoS protocols are in the spotlight as more and more high-profile blockchains, e.g., Ethereum, attempt to switch over from PoW protocols. This transition may be in the form of hybrid systems (PoW/PoS) [2] or complete transition to pure PoS implementations [3], [4]. The main reason behind this shift is to reduce energy consumption since mining is very computationally intensive. It is indicative that Bitcoin mining consumes more electricity than 159 countries of the world [5]. Nevertheless, the real power consumption could be far more, since many malware perform mining on infected devices without the users' knowledge [6].

While blockchains are considered very secure and privacy-safe by their very nature, this is not at all an accurate sentiment, as there are several attacks that may exploit them in various ways. Some of these attacks are somewhat technical and may only apply to specific blockchains. One should consider that blockchains are a technology which may be around for a decade, yet the bulk of the people who are using them have been only recently involved with them and in many cases treat them as a ''blackbox.'' Therefore, we argue that there is a need to shed light on various security aspects of blockchains, analyse the threats they are exposed

The associate editor coordinating the review of this manuscript and approving it for publication was Sharief Oteafy.

to, the attacks that can be performed, their expected impact as well as possible countermeasures.

The shift towards PoS, motivated us to investigate attacks targeting PoS protocols and focus more on one of the greatest threats against them which are *Long-Range attacks*. *Weak Subjectivity* and *Costless Simulation*; which will be discussed in Section II, amplify the impact that these attacks have in PoS protocols compared to PoW protocols. In essence, in a Long-Range attack, the adversary creates a branch on the original blockchain which may contain different transactions and blocks and overtakes the main chain. This branch is also referred to in the literature as *Alternative History* or *History Revision attack*. In the rest of this work, we will use the terms Long-Range, Alternative History, Alternate History, History Revision interchangeably.

Long-Range attacks are only relevant for PoS protocols. For PoW protocols, they are unfeasible due to the required computational effort for generating the previous blocks and outpacing the main chain. On the contrary, an attack on a small number of blocks is considered a *Short-Range* and targets at reorganising blocks of, e.g., few days up to a few months. As a result, Short-Range attacks have different incentives, execution methods, impact, and mitigation techniques which will be briefly discussed in Section 3.

While there is no formal definition of long range attacks in the literature we assume that long range attacks are attacks whose aim is to rewrite the history of a blockchain. To this end, rather than targeting towards recent transactions, long range attacks are forks of the blockchain which may fork it even from the genesis block. One could formally define long-range attacks as follows:

Definition 1 (Long-Range Attacks): Let $\mathbb{B}$ a blockchain whose main chain consists of n blocks, namely $(b_0, b_1, \ldots, b_n)$. A long-range attack is an attempt of an adversary to transform $\mathbb{B}$ into $\mathbb{B}'$ consisting of $k \geq n$ blocks $(b'_0, b'_1, \ldots, b'_k)$ forking from block f such that:

- $b_i = b'_i, \forall i < f$, and
- $\left| \{b_i \neq b'_i, i \in [f, \ldots, n]\} \right|$ is not negligible.

For the sake of clarity, in the rest of this work, we assume that long-range attacks forked from the genesis block (b_0).

Currently, there are several surveys in the literature covering various aspects of blockchain security, none of which covers Long-Range attacks. For instance, Grech *et al.* [7] present gas related attacks for Ethereum contracts. Bartoletti *et al.* [8] study Ponzi schemes on Ethereum. Atzei *et al.* [9] present a comprehensive overview of attacks on Ethereum smart contracts, and later Nicolic *et al.* [10] performed a large-scale study on available vulnerable smart contracts. Conti *et al.* [11] analyse the security and privacy issues of Bitcoin and de Balthasar and Hernandez-Castro [12] analyse Bitcoin laundering mechanisms. Li *et al.* [13] performed a systematic examination on the security risks to blockchain and provided an overview of many attacks to popular blockchains, however, their approach does not efficiently cover attacks on PoS protocols nor Long-Range attacks.

The rest of this work is organised as follows. In the next section, we provide an overview of the needed background notions for understanding blockchain-based networks, focusing on PoS protocols, and introduce the reader to some fundamental concepts that will be used to describe Long-Range attacks. Then, in Section 3, we describe some of the most well-known attacks against common consensus protocols. In Section 4 we introduce the basic Long-Range attack scenarios affecting PoS-based protocols, and in Section 5 we discuss various countermeasures to address these attacks and their effectiveness. Finally, the article summarises our contributions and discusses open issues and future research directions.

II. BACKGROUND

In the following paragraphs, we provide a brief overview of the basic concepts about blockchains, focusing on PoS protocols and their two main variants. We also introduce the reader to the concepts of Finality, Costless Simulation, and Weak Subjectivity that play key roles in Long-Range attacks.

A. BLOCKCHAINS

A blockchain can be conceived as a *distributed append-only timestamped data structure*. This way, blockchains provide a distributed peer-to-peer network enabling members that do not necessarily trust each other to interact verifiably without the need for a trusted authority [14].

In principle, blockchains store transactions between the participants which are signed by at least one of them. The transactions may involve physical and/or digital assets or even state transitions. Once a transaction is performed by a node; a participant of the blockchain, it is disseminated to its peers and added to the pool of transactions. Some nodes may have elevated privileges compared to others. The ones that are in charge of determining whether the transactions are valid, which should be kept in the blockchain and which should not, are called *full nodes* and group transactions into *blocks*. All blocks are appended at the end of the chain and are intertwined with the previous block by containing its hash as part of the stored data. The fact that a transaction is not included in a block does not mean that it is discarded. If the transaction is valid, it will be included in a future block along with others. This creates a chain of blocks that allows one to trace possible alterations. Apparently, this chain has an initial block that we refer to as *Genesis block*. Due to its nature, the Genesis block can be used to validate all appended blocks, and it is the only block that every node blindly trusts. Thus, when a new node is added to the network, it is always provided with the genesis block. Starting from this block, the node will be able to retrieve all of the currently published branches of that blockchain. In a pure PoS protocol, the Genesis block also contains information about the validator set [15].

Since there are many nodes and there is no central database, one may try to exploit the platform by performing multiple transactions, e.g., by tricking its peers that the current balance allows her to perform a transaction. In this regard,

full nodes are expected to come across transactions that are received with different order and possibly with colluding outcomes. Therefore, nodes must reach to an agreement, that we call *consensus*, on which transactions must be kept in the blockchain to guarantee that while there might be corrupt branches, users will be able to trace them and disregard them [14], [16]. The way that the nodes reach consensus is what differentiates most blockchain implementations. The ''traditional'' way, introduced with Bitcoin, is through what we call *Proof-of-Work (PoW)*. In this case, nodes have to prove that they spent a lot of computational effort by solving complex computational problems, e.g., hash-based crypto puzzles like hashcash [17] and in return, they expect a verification of their blocks. In *Proof-of-Stake (PoS)* protocols, the trust of the network is bounded on digital assets, the currency of each protocol. Instead of miners, PoS protocols use the term *minters* or *validators*. At any point of the protocol's operation, a static or dynamic set of validators stake their assets to the network to be capable of producing and validating blocks. Other consensus approaches include *Byzantine Fault Tolerance (BFT)* and its variations [18]–[20].

Evidently, the lack of a central and unique repository and complete overview of all the transactions communicated by nodes results to nodes having different branches of the blockchain until they reach a consensus. Therefore, nodes compete in having their branch as the main branch of the blockchain.

B. VALIDATING TRANSACTIONS IN PoW AND PoS

In principle, all participants of a blockchain network submit their transactions to a pool, the *pool of transactions*. These transactions must be verified and mined (in the case of PoW) to be added to the blockchain. To this end, in PoW, nodes commit their processing power in an attempt to gain a network reward. They can do so only if they are the first one to accomplish this task, while the rest of the nodes will see their time and computational power being thrown away. The network reward might be from the blockchain system itself, in the form of a *block reward* e.g. once a block is created and added to the main chain, the blockchain mints a certain amount of tokens and awards the miner for their hard work. Another form of reward can be from transaction fees. Due to the congestion of blockchain networks, users can add an extra cost to their transaction which will be given to the miners as a reward to have their transaction processed with a higher priority. Block rewards are not the de facto operation of a blockchain, e.g. while Bitcoin uses block rewards, the Ethereum network does not and miner rewards are solely based on transaction fees. It must be noted that even though the Bitcoin network uses block rewards, transaction fees can also be used.

In both consensus protocols, miners have complete control over their block and use the transaction pool to fill up their blocks with transactions and may prioritise the transactions as they see fit. In addition, the block size is finite, making this prioritisation usually based on the transaction fees. Therefore, in a blockchain system, a miner or validator selects some transactions from the transaction pool and adds them to their block. While in PoS systems the validator will only have to publish the block, in PoW systems the miner has to go one step further and solve a hashing puzzle. The willing miner will have to pay that cost; work for them, not knowing whether this work will be in vain or result into a monetary reward.

As already discussed, the task is usually a strong cryptographic or mathematical problem whose difficulty changes dynamically based on the overall network computational power to keep a stable pace of block generation (an average of 1 block per 10 minutes for Bitcoin). For instance, in Bitcoin, the challenge is to find a hash below a threshold. More precisely, one creates a block where the block header consists of the hash of the previous block, the network difficulty, a timestamp, a block version number, a merkle proof of the transactions she believes that should be added in the blockchain, and a nonce. By using different nonces, one tries to find a hash below the threshold that was set by the network difficulty. Once a node finds such a nonce, she disseminates it, and the block is added in the blockchain once the other nodes verify its validity.

In PoS blockchains, to generate blocks users will have to be part of the staking system as *validators*. Network users who want to participate in the staking system have to purchase tokens and stake them. The staking system is usually weighted, and validators with more stake in the system have more voting power consequently. In PoW systems the cryptographic puzzle is there to define the rate of block generation, in PoS systems this rate is defined with time intervals, e.g. every 10 minutes one validator is elected to create a block. In general, the election of the validator is made through a pseudo-random generator where candidate nodes with more stake are more likely to be elected to validate blocks.

Once the validator has concluded on the transaction selection for the block, a block is *forged* containing all the approved transactions. The block is then added in the blockchain or delegated to other validators for approval, depending on the type of proof of stake protocol in use.

For more details, the interested reader may refer to [21] and [22].

C. PROOF OF STAKE PROTOCOLS

PoS as a concept, contrary to the Bitcoin-like PoW protocols, does not rely on hashing or computing power to distribute ''ownership'' among peers on the network. As previously mentioned, it rests upon the idea that ''ownership'' between peers of the network is attributed to their ownership of ''stake'' in the form of the native currency (usually called coins), recognised and supported by the blockchain network in question. These coins grant them the ''voting power'' on the block generation and/or verification of the underlying network. As the concept of PoS evolved via various iterations of protocol implementations and networks, one major question was constantly pressuring developers and researchers to

answer, to ensure the long-term viability of PoS similarly to what PoW-based networks have managed to achieve. This question is:

> *Is it possible to achieve the same level of security (especially around consensus and voting) as that which Bitcoin-like networks can achieve, without depleting physical resources, which by definition are scarce within any such network?*

At the same time, first implementations of PoS-based protocols were based on naive assumptions and without further safeguards and subsequently were plagued by a number of potential adversarial attacks which we briefly explore in later sections of this manuscript.

Researchers poured significant resources into tackling the struggles mentioned above and eventually came up with two dominant approaches to secure consensus protocols, namely Byzantine Fault Tolerance based PoS and Chain-based PoS algorithms. In the next sections, we will explore the basic defining properties of PoS-based blockchain protocols, the ideas behind the two dominant approaches, their advantages and disadvantages regarding security and secure consensus protocols.

D. THE CONCEPT OF FINALITY

Before we discuss in detail about the two dominant approaches of PoS protocols, we first need to introduce one more concept that is directly linked with the differences in the consensus approach between the two which is the concept of transaction and block *finality*. Finality in blockchain-based transaction platforms essentially describes the guarantees that are provided by the blockchain that well-formed blocks that have been committed to the blockchain will not be revoked at a future point in time and therefore the included transactions have been finalised and can be trusted.

In traditional PoW-based systems, such as the Nakamoto consensus implemented in Bitcoin, this type of finality is not immediately guaranteed, but instead relies upon the concept of *Probabilistic Finality*, which is commonly associated with chain-based protocols (regardless of PoW or PoS). This type of finality is achieved by consistently reducing the likelihood of a transaction or block being altered or reverted, the deeper this block is located within the blockchain. For instance, when using Bitcoin, most major Bitcoin transacting platforms usually wait for at least five blocks to be generated on top of the original transaction block to validate and accept a transaction as trusted. This approach, of course, introduces other problems, as six blocks (the original block and the five ones afterwards) in the Bitcoin chain usually take a prohibitive amount of time to be generated for a network that aims to support real-time transactions, as it may span to several minutes or even hours.[1] While the same principle of Probabilistic Finality applies for Chain-based PoS networks, the drawbacks are far less as there is a minimal commitment of computing resources to generate a block and therefore the required generation of a 6-block chain to validate transactions happens substantially sooner.

BFT PoS-based networks, however, take a very different approach to transaction finality, in an attempt to ensure near real-time transaction finalisation and acceptance on the blockchain. This type of finality is usually referred to as *Absolute Finality*. This type of finality is achieved by definition in BFT-based networks, according to their basic consensus principles, which will be explored in detail within the following paragraphs. The idea is that a randomly chosen validator proposes a transaction block. Then, the rest of the validators vote on the acceptance and commitment of the block to the chain, and once a sufficient supermajority of validators voting for the block is achieved, the block is *irreversibly* committed to the blockchain. There are slight variations of the Absolute Finality implemented in various BFT-based PoS blockchains, while the result is to always ensure and guarantee immediate and absolute finality of committed blocks without running the risk of a future majority reverting or altering any parts of the blockchain.

In reality, absolute finality is almost impossible to achieve in PoS protocol implementations without sacrificing the availability of the network itself. In the ideal scenario where absolute finality could be achieved (a transaction or block could never be reverted), it would require all registered validators of the network to come to a single agreement for a block validation. To that end, they would all have to vote for the same specific block on the main chain and no other blocks on any forks. The strict nature of this requirement would result in the network failing to validate a block in the majority of cases, as connectivity issues, node desynchronisations and outages for even a single validator would halt the validation process. To overcome this limitation, while still providing near-absolute finality on validated blocks, most BFT-based PoS networks employ the concept of ''economic finality,'' which will be discussed in detail in Section 5.

E. BYZANTINE FAULT TOLERANT-BASED PROOF-OF-STAKE

The well-known problem of Byzantine Fault Tolerance (BFT) [23] is very relevant for distributed computing as it describes many of the challenges that nodes face when trying to achieve trusted and secure consensus over an untrusted network. In such a hostile environment, apart from inherent errors of the communication channels, a group of nodes can be malicious, attempting to subvert the procedure, as long as the trustworthy nodes exceed a specific supermajority. Specifically on PoS protocols, BFT solutions were first introduced in the blockchains with the implementation of Practical Byzantine Fault Tolerant (PBFT) algorithm of Castro *et al.* [24].

In these implementations, the network, following a round-robin approach, randomly designates validator nodes as leaders to propose new blocks. It then requires a supermajority exceeding 2/3 of the total validators on the network to agree on accepting the proposed block and instantly integrating it as part of the blockchain. This way, they implement

[1] https://www.blockchain.com/en/charts/avg-confirmation-time

near-absolute finality (as described above) after a 2/3 consensus is reached. This actually means that BFT PoS protocols can tolerate a fault rate of up to 1/3 of the total participating validator nodes at any given block validation (independently of whether these faults are attributed to network errors or malicious behaviour), ensuring that a transaction signed by more than 2/3 of validator nodes is finalised and will not be reverted in the future under any circumstances.

Briefly, in a practical BFT implementation on a blockchain network the four primary phases of a BFT consensus protocol are the following:

1) A client node sends an operation request (transaction) to the currently designated leader node.
2) The leader node also broadcasts the request to all other participating validator nodes.
3) Each node independently executes the operation and sends its reply to the initiating client node with the result of the operation.
4) The result that is transmitted from more than 2/3 of the total validator nodes is considered the final result of the operation.

PoS protocols implement a BFT approach, always ensuring that the network is in a consistent state. The latter, of course, may result in temporary unavailability or high network delays while recovering from occurring faults, when more than 1/3 of the validators are unresponsive. One such de-facto state engine that implements and offers BFT PoS blockchain consensus is the Tendermint state-machine and more specifically the Tendermint Core implementation [25].

To summarize, BFT-based PoS consensus protocols achieve the following properties:

- Fault tolerance for faults up to 1/3 of the total number of validators on the network per block voting/finalization cycle
- Requirement for synchronous validator nodes (as finalization happens on the spot)
- Prioritisation of consistency over all other properties for the blockchain
- Consensus safety
- The instant and absolute finality of blocks
- Support for both public and private chain models

F. CHAIN-BASED PROOF-OF-STAKE

Chain-based PoS frameworks took a more conservative approach in defining secure consensus protocols by trying to emulate the well-established PoW paradigm of achieving consensus. To this end, they link the newly generated and committed block to the previous blocks via hash-linking to a parent block belonging to the previously established longest chain (which is usually accepted as the valid chain). This linking is considered unique and one-way and can be used to validate all subsequent blocks of the longest chain by retracing the linking from the first block of the chain all the way to the current block. The validation is achieved using standard cryptographic signature concepts.

As previously explained, on the finality section, there are no guarantees during linking of a newly generated block about its finality, instead the finality is achieved probabilistically after enough child blocks have been linked to the block in question and the block is ''deep enough'' in the chain that it is considered ''probabilistically finalised.'' In reality, a chain-based PoS blockchain is never considered consensus safe, and there is no guarantee that the blockchain will always maintain a safe state. On the contrary, priority is set on the availability of the network, which can continue operations even if only a fragment of the validators are still active. To ensure that the blockchain will not reach a finality bottleneck, the network must guarantee that block generation happening at the head of the chain is always slightly faster than the progression of block finalization, to ensure that the ''deepness'' requirement for safety is consistently reached for committed blocks.

It is worth noting a fundamental difference here. While PoW consensus protocols require a fixed number of block iterations before they assume finality (6 for Bitcoin), Chain-based PoS protocols usually allow for customization of those finality threshold for each validator node [26]. This ensures that a certain amount of safety can be chosen individually, while the network still maintains certain low overhead in performance, usually encountered in PoW protocols.

Concluding, Chain-Based PoS consensus protocols achieve the following properties:

- availability of the network even during high fault rates,
- customizable safety thresholds and
- accommodate asynchronous validators with guaranteed individual safety thresholds.

G. COSTLESS SIMULATION AND WEAK SUBJECTIVITY

The main advantage of PoS protocols can also be considered their Achille's heel since it constitutes the primary source of the attacks that can be launched against them. As already discussed, PoS protocols do not require any substantial computational effort for the validators. While this minimises the energy consumption, it also implies that an attacker does not have any cost to launch an attack. We refer to this concept as *Costless Simulation* or as it is more widely known as *Nothing at Stake*. Practically, the validator has nothing to risk when making consensus decisions, therefore, his optimal behavior is to participate in as many forks of the chain as possible as he is not aware of which one will become the main one. However, this hinders the stability and the goals of the main chain. An adversary may create an alternative branch to the main chain of a PoS-based blockchain starting at any point that he wants without having any actual cost. Clearly, this would be impossible in PoW-based blockchains as it would require an enormous amount of time and computational power to accomplish.

The term *Weak Subjectivity* is used to describe a problem that affects two type of nodes. The first one is new nodes of a blockchain network and the second one is nodes which are brought back online after a significant amount of time being offline. Clearly, in both cases the nodes due to lack of

synchronization they are not able to have a concise picture of the main chain of the blockchain since they lack ground truth about the evolution of the blockchain during the pass of time. Therefore, weak subjectivity does not affect online and synchronized nodes. Starting from the genesis block, nodes that just joined the network can retrieve all currently published branches of the blockchain. Unfortunately, a new node will not be able to distinguish the main branch of the chain immediately. The latter limitation also applies to nodes that have been offline for an extended period. At some point in the past, these nodes were aware of which branch was the main chain; however, after this dormant period, they are not aware of it. Since online nodes monitor the blockchain in real-time, they cannot be tricked into accepting a different branch as the main chain unless this branch legitimately becomes the main chain. Thus, weak subjectivity is the problem that arises when nodes come online, there is not a single branch of the blockchain, and they cannot determine which one is the main.

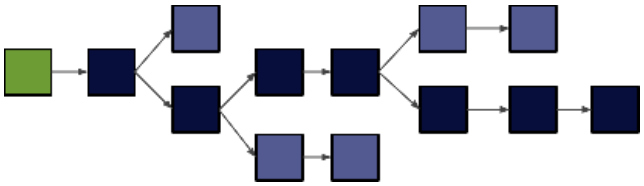

FIGURE 1. Sample blockchain. The green block is the genesis block, light blue denotes the branches, and the dark blue denotes blocks of the main chain.

As shown in Figure 1, a typical blockchain snapshot has many different branches. Some of them are longer than others, and all of them are candidates of becoming the main chain.

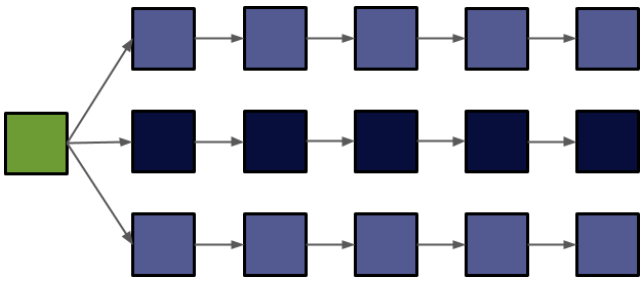

FIGURE 2. Blockchain example where multiple branches have the same length.

As a result of the above, any node joining the network will be presented with multiple branches of the blockchain, many of which may have the same length as shown in Figure 2, yet the node will not be able to determine the main chain and may be tricked into accepting a malicious one.

III. CONCERNS AND ATTACKS ON CONSENSUS PROTOCOLS

In trust-less systems such as the blockchain, security is a top priority. It is expected to be faced with attacks that will attempt to undermine the consensus and steal funds from its users. This section presents some general issues of blockchains and some of the most well-known attacks against consensus protocols.

A. DOUBLE SPEND

Double spending is one of the problems that blockchain technologies attempt to solve since their very inception [1]. Most, if not all of the attacks in the blockchain, aim to perform a double spend at some point in their execution. In this attack scenario, an attacker attempts to spend the same currency at least two times, hence double-spend. This attack is definitely not possible in the physical terms of currency. It is not possible to buy a resource from one vendor and then spend the exact same coins to another vendor. The attacker attempts to perform a transaction, wait for the merchant to approve it, and then revert it and spend the same currency in another transaction. In blockchains, this can be achieved by presenting a conflicting transaction possibly in a different branch. BFT systems [19] with the use of absolute finality are considered to be robust against the double spend problem.

B. SYBIL ATTACK

Sybil attacks [27], [28] usually refer to peering networks affecting members of the network with equal rights. Even though these attacks can be executed against blockchain protocols, with the use of proper consensus algorithms these attacks can be successfully mitigated.

In a Sybil attack an adversary uses multiple identities to disrupt or misguide a network decision or opinion. In blockchains, this could create many issues such as in finalising a block, branching the blockchain, electing validators etc. For a PoW system, the attacker would have to own various identities with sufficient computational power to have substantial influence in the system. The consensus protocol directly defeats this argument as the attacker would have to split his computational power in smaller chunks. This would not be a benefit for the attacker and would not amplify his computing power. Instead, the attacker's computing power will remain the same, if all identities mine the same block, or diminish, if each identity mines a different block. In either case, the attacker will have to own a substantial amount of computing power which will cost her the same as having all of it concentrated in one entity or split in multiple ones.

Similarly, in PoS systems, the attacker is directly countered by the consensus protocol. Validator election process and voting power are weighted based on their ''stake.'' The attacker will have the same voting power in case one identity owns all of the attacker's stake, or it is split into multiple identities.

C. RACE ATTACK

Race attack [29], is usually considered as an implementation issue and relies on the existence of probabilistic finality. In blockchains where finality is not absolute, a number of confirmations must be reached before considering a transaction finalised. A recommended number is usually known to

the network, but depending on the transaction a more empiric number can be used at the discretion of the merchant/user. Transactions handling a considerable amount of funds might require a greater number of confirmations before being accepted as valid. It is possible that a service is not configured correctly and does not wait for the recommended number of confirmations for that specific blockchain. In that case, a double spending attack is possible. An adversary would mine a fork of the blockchain till it satisfies the weak requirements of that service, retrieve the goods of that ''weakly''-verified transaction, and then divert his computing power from that branch to the main branch. This way, the ''weakly''-verified transaction will be soon lost in a branch of the blockchain and will not be valid for any node following the main chain. This attack is mitigated by BFT-based Proof of Stake protocols with the use of absolute finality.

D. BRIBERY ATTACK

Bribery attack [30], also referred to as *Short-Range attack* relies on bribing validators or miners to work on specific blocks or forks. By doing that, the attacker can present arbitrary transactions as valid and having dishonest nodes paid to verify them. By paying them an amount equal to or more than the block rewards (in case the block is reverted by the network), it provides an incentive high enough for miners to work on the attacker's blocks or chain. This case of bribery attacks also known as *P+epsilon attack*[2] states that it is possible to bribe users without having to pay them, as the system will award the bribe to the dishonest nodes by making that branch the main chain. For these cases, the attacker faces a more significant problem as in case the malicious branch is reverted for some reason (attacker cannot continue the bribe, dishonest nodes stop working on that branch) the attacker would have to pay an enormous amount of bribes as the bribes will accumulate for every maliciously minted block.

In PoS systems, these kind of attacks are feasible and can be expanded to the nothing at stake problem. In both cases, PoS tackles this issue by either enforcing a slashing condition [2] or by releasing violators from their position [15].

E. LIVENESS DENIAL

Liveness Denial is a form of Denial of Service attack in PoS protocols. In this attack, some or all of the validators decide to take action and purposefully block transactions by stopping publishing blocks. By avoiding to perform their validator duties, the blockchain will come to a halt as new blocks would not be able to be validated and published in the blockchain.

A liveness requirement [2] which slowly drains the stake of inactive validators will ensure that even if the majority of validators are either offline or performing a liveness denial attack, they would not compromise the network. In cases where liveness cannot be assessed, the community will be able to decide (off-chain communication) to fork the blockchain and remove the inactive validators. In all cases, validators who conduct this type of attack jeopardise their position in the network as validators and their stake if a slashing condition exists [2].

F. CENSORSHIP

Censorship in blockchains is big a thorny issue that has sparked many discussions as it can be considered an attack or a feature simultaneously, depending on the nature of the blockchain. Validators have control of which transactions will be added in a block which gives them the power to blacklist certain addresses. Transactions lie in the transaction pool, and validators take transactions and add them to their soon-to-be-published block. Validators might decide to remove some transactions from their blocks. In the scenario of a single validator performing the censorship, some transactions might be delayed or be invalidated due to time constraints. The danger of censorship becoming more real is amplified once the number of validators performing this attack increases.

Liveness requirements, can ensure the eventual process of transactions [15] and eliminate censorship on the blockchain. In addition to that, the protocol can punish nodes which do not create blocks in a protocol-defined order [31]. In another more effective solution, *Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs)*, can be used to hide the identity of the transaction sender [32].

G. 51% ATTACK

51% attacks are a threat to any consensus protocol. In PoW systems, the entity which controls the majority of the hashing power at a specific timeframe can have complete control over the blockchain, for instance, she can fork the main chain and start mining on her branch. Slowly and steadily, the adversary will be able to outpace the main chain and have her branch take its place. Since in PoW protocols block generation is probabilistic, there are many chances to have conflicting branches, which diminish as we move past the 51% barrier. Therefore, the adversary will have the main chain, but branch reversions will often happen at that point.

It should be noted that due to the probabilistic rewards of PoW blockchain, miners tend to unite computation powers with the use of *mining pools*. Mining pools while ensuring that nodes with low computing power can get rewards from block rewards, they can also unwillingly lead to 51% attacks. In cases where one mining pool reaches high levels of computational power, the miner community shifts and moves to competitor mining pools to stabilise the network and avoid accidental 51% attacks.

In PoS protocols this attack is still viable but with a slightly different impact. It is possible for one validator or a coordinated set of validators to own more than 34% (for BFT PoS) of that blockchain's stake. In this case, a majority attack can impact the blockchain by performing finality reversion, where an already finalised block is being challenged by finalising another competing block, causing Liveness Denial or Censorship.

[2]https://blog.ethereum.org/2015/01/28/p-epsilon-attack/

Currently, there is no concrete solution to this problem. When a 51% attack occurs, it will usually be notified by the miners, and a community-driven fork will take place to re-establish the honest chain as the main chain.

H. SELFISH MINING

In *selfish mining* attacks [33] also known as *block withholding*, the adversary mines blocks in their own fork of the blockchain without publishing them to the network. Once the attacker has computed a desired number of blocks, they are released to the network and aim to revert the main chain to that of the attackers. The purpose of this attack can be two-fold; a) disruption of the network by wasting resources of honest nodes and b) increase the rewards collected by the dishonest nodes.

The former incentive of the attack affects purely PoW blockchains. A PoS blockchain would not be disrupted by this incentive. The latter incentive affects both protocols and can be mitigated by applying slashing conditions [25], [34] or by removing violators from their position of power which will cause them to forfeit future rewards [15].

I. OTHER ATTACKS

Grinding attack, also known as *precomputation attack* [4], is an implementation specific issue and affects PoS systems. By exploiting the lack of randomness in the slot leader election process, a slot leader is capable of manipulating the frequency of them being elected in subsequent blocks. This issue can be solved by enforcing randomness to the process and minimising or even eliminating influence factors of this process which are controlled by the validators [4], [15].

In another implementation specific issue, the *Coin Age Accumulation* attack affected Peercoin [3]. In the early version of the protocol, the user's stake would accumulate more weight the more time it was staked without having any time restrictions. Given enough time, an attacker would have accumulated enormous amounts of stake in the system which would allow them to take over the network. The coin age mechanism worked as an amplification technique to the stake of its validators. As a mitigation technique, the coin age mechanism was capped at a certain amount of time or was completely removed [4].

Other attacks in the blockchain include the *Eclipse attack* [35] where an adversary attempts to obstruct message delivery of nodes in the peer-to-peer network level causing nodes to work on a corrupted or distorted snapshot of the blockchain.

IV. LONG-RANGE ATTACKS

As already discussed, a Long-Range attack is an attack scenario where the adversary goes back to the genesis block and forks the blockchain. The new branch is populated with a partially, or even completely, different history than the main chain. The attack succeeds when the branch that is crafted by the adversary becomes longer than the main chain, hence it overtakes it. Long-Range attacks fall into three different categories, namely *Simple*, *Posterior Corruption*, and *Stake bleeding*.

In some sense, Long-Range attacks in PoS protocols are related to selfish mining attacks of PoW protocols as the attacker in both cases is adding blocks that she keeps secret. Obviously, selfish mining attacks cannot go back to the genesis block of PoW protocols as the computational effort needed is prohibiting, therefore, the impact that they may have is limited. Nevertheless, both attacks fork the main chain and try to append forged blocks where the attacker potentially includes different transactions.

Starting with the simplest case of Long-Range attacks, we will build up to the more complex scenarios. In our examples, we consider a validator pool with three validators, that we denote as Alice, Bob, and Malory, where the latter is the adversary. For the sake of simplicity, we consider that all of them own the same portion of the system's stake. Therefore, each owns 33,3% of the stake.

In Simple attacks, we consider a naive implementation of the PoS protocol in which the nodes do not check the timestamps of the blocks. Therefore, when a PoS protocol is executed, every validator will have the possibility to validate blocks, as shown in Figure 3.

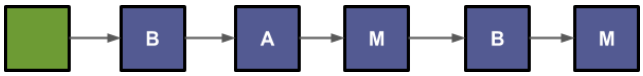

FIGURE 3. A snapshot of a blockchain which has 3 validators, Bob, Alice, and Malory, each of which has equal chance of getting elected and validating a block.

Once Malory initiates a Long-Range attack, she creates an alternative branch of the blockchain. Malory forks the blockchain starting from the genesis block, and starts minting a branch which may contain different transactions.

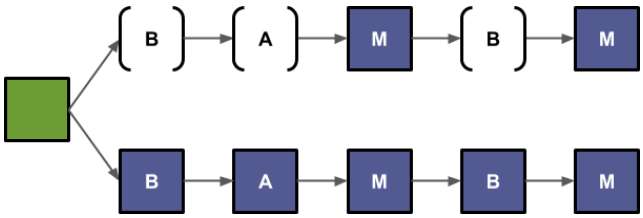

FIGURE 4. The lower branch is the main branch while the upper is Malory's. Malory has the same chance of being elected in both branches. Forfeited blocks are denoted with a parenthesis. Currently, Malory's branch has 2 blocks.

As seen in Figure 4, since the validator information is located inside the Genesis block, Malory would not be able to produce blocks at a faster pace than she would in the main-chain. Hence, Malory produces blocks at the same rate. Because of these constraints, Malory has to produce blocks ahead of time to advance her branch and overtake the main-chain as seen in Figure 5.

To this end, Malory would have to forge timestamps. Since Malory is the only active stakeholder in the branch that

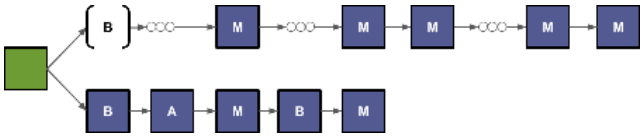

FIGURE 5. With the use of triple dots we denote multiple forfeited blocks. In Malory's attempt to compete with the main-chain, she computes blocks ahead of time. In this snapshot, both branches have the same length.

she forked, she can manipulate them according to her will. As a result, in implementations where nodes do not take into consideration timestamps, both branches will be valid, and nodes would not be able to spot Malory's forgery.

A. POSTERIOR CORRUPTION

Let us now consider the scenario where Malory cannot forge the timestamps of the blocks. To alter the history of the main chain, she must generate a higher number of minted blocks in parallel to the main-chain. However, she is bounded by the fact that she has a fixed chance of producing blocks.

To increase her chances of competing with the main-chain, she would need to be able to forge the blocks of another validator as well. If for instance, Bob agrees in attacking the main-chain, then the attack is rather straightforward, as both of them would have far more than 50% of the stake. However, this is not necessarily the case.

To clarify the attack, we need to introduce the concept of *validator rotation*. Due to the nature of blockchains, the requirements and expectations we have from them, we must not have any static component or entity beyond the genesis block. Validators are no exception to this rule, hence, not only they cannot be considered static, but validators have to rotate in order to have a fair system. Moreover, validators should have the option to retire, and the system should rotate or remove them under certain conditions, such as a set period of time or dishonest behaviour. Finally, when considering an entity as benign and trustworthy, this does not mean that the entity will remain so after leaving the system. One should consider that the validator's incentives may change over time or their system might be compromised.

For the sake of simplicity, we consider that Bob decides to retire after validating the first n blocks of the main chain. Bob removes his stake, cashes out, therefore, he is not part of the blockchain any more, but his blocks are. While Bob was a validator, he might have used security best practices for storing and managing his private keys. However, once Bob does no longer own any stake on the platform, we argue that the security of his private key for validating his blocks is not a priority for him.

Note that even though Bob cannot sign new blocks since he is no longer a validator, he (or whoever has access to his keys) can sign the first n blocks of any branch of that blockchain. Practically, this means, that having access to Bob's keys one may forge the blocks he validated. One must also consider that Bob is no longer part of the system and therefore there is nothing at stake for Bob, so he does not have any disincentive not to perform an attack against the blockchain. In practice, an attack by Bob would be Bob signing the first n blocks on another chain of the blockchain for or with an attacker.

Based on the above, there are two possible scenarios for an attack, that we name *Posterior Corruption*. Either Malory hacks Bob and steals his private key, or Malory bribes Bob, and he joins the attack. In either case, Bob's private key is known to Malory, who can sign valid blocks, masqueraded as Bob, increasing her chances of outpacing the main-chain.

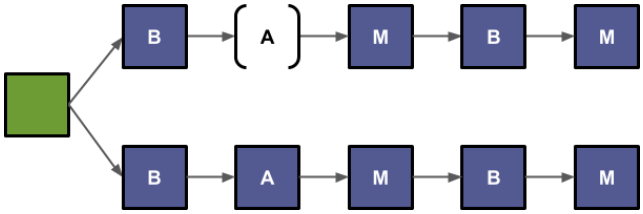

FIGURE 6. The lower branch is the main chain while the upper branch is Malory's branch which competes with the main chain to overtake it. In this figure, Bob joins Malory's Long-Range attack. Now the alternative (upper) branch is much more competitive and the chances to overtake the main-chain are increased.

Since Bob is participating in the attack; directly or indirectly, he will produce a block in Malory's branch every time he is elected as a block validator. As illustrated in Figure 6, Bob's blocks in Malory's branch are no longer empty, and the malicious branch has better chances of competing with the main chain.

B. STAKE BLEEDING

Let us assume again that Malory, as validator wants to perform another type of attack. As in the previous cases, Malory has forked the main chain and hidden her branch and plans to publish it when she outpaces the main-chain. Clearly, when the branch occurs, Malory has the same chances of being elected as a slot leader in all branches of the blockchain. See Figure 7.

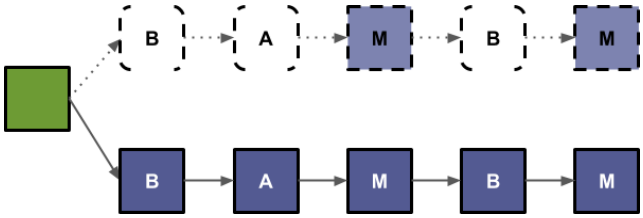

FIGURE 7. The lower branch is the main chain while the upper branch is Malory's attempt to create a new main branch. Malory's branch (dotted branch) is mined and stored locally and not published at the moment.

Malory, to increase her chances of achieving this attack, starts to stall the main chain. The concept is that by stalking the main chain, she will have the necessary time to produce blocks and outpace the main chain. Depending on the share of the total stake that Malory has this could result in a *Liveness Denial attack*. In what follows, we assume that validator rewards are compound on their stake.

To launch the attack, every time Malory is elected as a slot leader in the main-chain she skips her turn, forfeiting her position, as illustrated in Figure 8. However, this does not mean that another validator will take her place, instead, for that slot/epoch, no new block will be added to the blockchain.

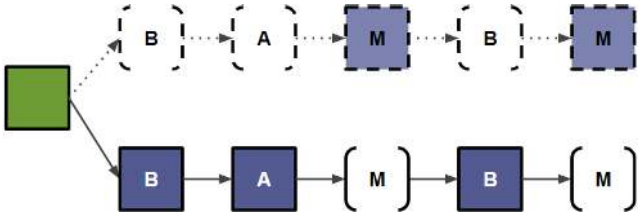

FIGURE 8. While Malory mines her own branch (upper branch), she continues to be part of the main branch (lower branch). Malory forfeits her blocks in the main chain to stall the main chain's growth and gain a competitive advantage on the alternative branch.

As a result, Malory will not get any rewards from the system, and her stake will gradually decrease. Eventually, all other validators will publish blocks and get the block rewards/transaction fees.

Simultaneously, in her own branch, Malory will be the only validator publishing blocks. Therefore, Malory will publish a block whenever she is given a chance. In this attack, Malory will try to increase her stake in every possible way. While performing this attack, Malory is behind the main chain, and thus she is capable of monitoring the main chain's actions, e.g. which transaction it chooses to add to each block. In addition to that Malory has access to the transaction pool from which she can choose any transaction located in it to add it to her blocks as long as these transactions are valid in the context of her branch. To this end, she will copy transactions from the main chain and or the transaction pool and publish them into her own branch. The goal of the latter is to maximize the number of transaction fees she will get and use them to increase her personal stake. Malory might prefer to copy transactions already located in the main chain to raise less suspicions from the network monitors.

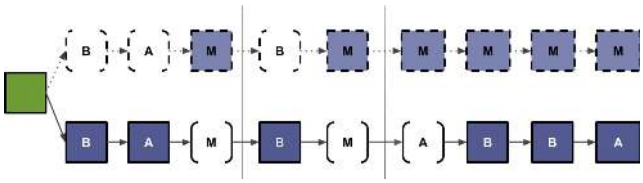

FIGURE 9. Malory's stake gradually increases in her branch (upper branch) and is elected more often as a block validator while she keeps losing stake in the main chain and as a consequence elected less often for block validator.

By stalling the main chain and publishing as many blocks as possible in the alternative chain she forked, Malory will eventually get the majority of the stake on her branch. Therefore, Malory's stake will start expanding faster than the main-chain, as illustrated in Figure 9. Once her branch outpaces the main-chain, Malory will make one last transaction in which she redistributes the stake to other validators and she then will publish her forged branch.

It should be highlighted that the attack is rather slow. According to [36] approximately a six-year worth of blockchain history for an attacker with 30% stake was needed to perform this attack successfully. On the contrary, with Costless Simulation, creating a branch with that amount of blockchain history can be considered instant.

V. COUNTERMEASURES

Various mitigation techniques have been proposed in the literature to counter these attacks. Even though all of them offer some protection against Long-Range attacks, none of them individually can be considered a complete mitigation technique. Therefore, a combination of these solutions is needed to protect against this type of attacks.

A. LONGEST CHAIN RULE

The *Longest Chain Rule* is the most straightforward technique to defeat weak subjectivity. For PoS protocols this technique can be considered the default option and is always used in conjunction with others. It should be noted that PoW protocols can counter weak subjectivity by using only this technique. In practice, for PoW protocols, the longest chain is considered the chain with most computational work.

In principle, according to this Longest Chain rule, the main chain is the branch with the most blocks, see Figure 1. While unusual, chain reorganisation can happen from time to time and in which blocks will be reorganised as the blockchain will accept a different branch as the main chain. When a branch of the chain becomes longer than the main chain, the blockchain restructures itself and holds the longest branch as the main chain. The concept behind this rule is that the network will recognise as the main chain the branch where most work has been made, as, e.g., in the case of PoW two miners may have almost simultaneously found a valid block. Note that even though the transactions in the other branch do disappear, they are not entirely destroyed. They are shifted to the pool of unconfirmed transactions where they might be placed into a subsequent block. Therefore, double spending issues can be easily traced and avoided.

B. MOVING CHECKPOINTS

Moving checkpoints or simply *checkpoints* is a mitigation technique used by almost all PoS protocols. Its simplicity and ease in implementation made it one of the first mitigation techniques to be enforced in PoS powered blockchains, after of course the longest-chain rule.

The idea behind moving checkpoints is that there is a quota on the latest n number of blocks of the chain which can be reorganised. Depending on the protocol, the block quota can change. In the case of Peercoin [37] the quota is limited to one month's worth of blocks and for NXT [38] it is the equivalent of a few days or hours. See Figure 10.

This defense mechanism partially mitigates Long Range attacks as it allows for some block reorganisation to happen.

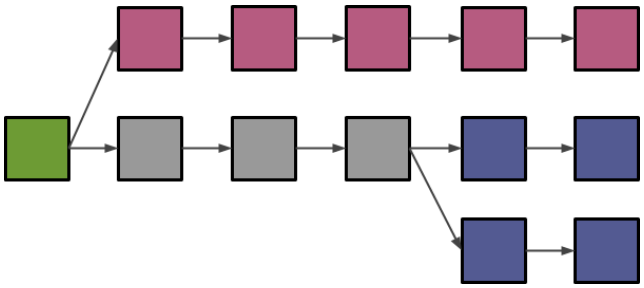

FIGURE 10. Snapshot of a blockchain using moving checkpoints with a factor of 2. Only the last two blocks can be reorganised. Grey blocks denote the truly immutable part of the blockchain. The magenta branch is invalid as it attempts to alter the grey blocks.

Nonetheless, with Moving checkpoints, Long Range attacks are downgraded and now fall under the category of Short Range attacks as the reorganisation does not start at the Genesis block. With the use of moving checkpoints, the main chain becomes truly immutable up to the latest n blocks.

C. KEY-EVOLVING CRYPTOGRAPHY

As already discussed, in Posterior Corruption attacks, retired validators, regardless of the fact that their keys are no longer used or valid, they can be used by an adversary to sign older blocks in the blockchain.

To counter this threat, David *et al.* [39] proposed the use of key-evolving cryptography [40], [41]. The concept behind key-evolving cryptography and more precisely the key-evolving signatures (KES) is that the lifetime of the key is divided into epochs for which a different private key is used, yet the public one remains the same. In this regard, there is a key update algorithm to derive the new private key from the previous one. Therefore, the epoch that the signature was issued becomes an integral part of the whole signature. As a result, even if a key is leaked it cannot be used to re-sign older messages.

Based on the above, the approach of David *et al.* is to force the slot leader to sign a block with a KES scheme so that the used key becomes immediately useless for an adversary. The use of continuously evolving keys and without the capability to recover an older version of the signing key renders the attack obsolete. If Bob decides to sign old blocks of a different branch, he will not be able to use the key he had back in the Genesis block, since it would have already evolved by his signing for another branch.

D. CONTEXT-AWARE TRANSACTIONS

As already discussed, every block in the blockchain contains the hash of the previous block in its header. This is used to identify the branch in which a block is placed and link them. With *Context-Aware Transactions*, we are moving this link one step further by including the hash of a previous block inside a transaction [42].

This way, we associate a transaction with a specific block in a particular branch of the blockchain. Since a transaction contains a historical reference of the blockchain, it cannot be copied to another branch and still be valid, unless the historical reference exists in that branch as well. See Figure 11.

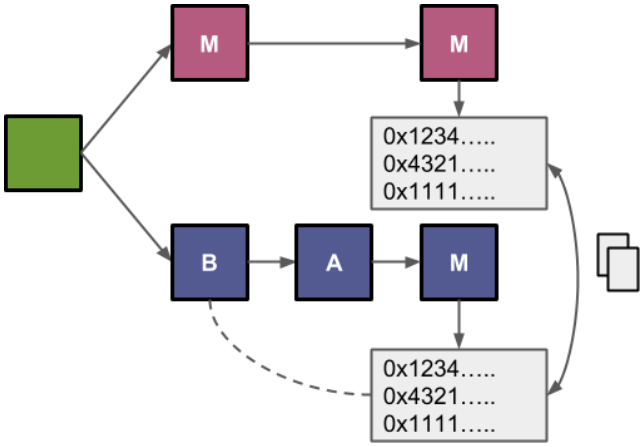

FIGURE 11. Malory copies transactions from the purple branch to progress her chain (magenta) but the transactions are linked to Bob's block. Malory's branch (magenta) will be rejected by honest nodes as inconsistent.

Recently, Coleman proposed the use of *Universal Hash Time* which is a type of Context-Aware Transactions [43]. Universal Hash Time uses hashes of time to establish transaction references bound to specific points in time.

With context-aware transactions, an adversary would not be able to copy transactions from the main chain to her minted branch. While it does not eliminate the attack, it introduces a substantial obstacle as the attacker is forced to create an entirely different history which significantly decreases its impact. Therefore, it can be considered an effective mitigation technique against Long-Range attacks.

E. PLENITUDE RULE

A more sophisticated countermeasure for Long-Range attacks is the *Plenitude Rule* which was first presented among other mitigation techniques as part of the PoS-based protocol Ouroboros Genesis [44]. This chain selection rule is based on detecting the sparsity and density of blocks in conflicting branches.

Clearly, an adversary cannot manipulate her stake in the main chain. Thus, an adversary with a starting stake of 20% will have the same stake on all branches of the blockchain, assuming that all branches derive from the same starting block, as seen in Figure 12. In Long-Range attacks, the adversary starts with a small chance to produce blocks and as block rewards are gained, and stake is accumulated more and more stake will add up for the adversary. The more stake a validator has, the more chances she gets to produce blocks.

Before the malicious validators accumulate enough stake to accelerate their block generation rate, the rest of the validators, the honest ones, according to our example will be elected for generating 80% of blocks. Based on the latter, it is evident that one branch will produce only a portion of the blocks the

TABLE 1. Countermeasures, attacks they address, and protocols that implement them. Protocol legend: ⋄ Ouroboros Genesis [44], ⋆ Ouroboros Praos [39], ∗ Ouroboros [15], ∘ Casper FFG [2], • Tendermint [25], ■ Peercoin [37], □ NXT [38].

	Simple	Posterior Corruption	Stake Bleeding	Protocols
Timestamp	✓	✓	✓	⋄ ⋆ ∗ ∘ •■□
Longest Chain	✓	✓	✓	⋄ ⋆ ∗ ∘ •■□
Moving Checkpoints	✓	✓	✓	⋄ ⋆ ∗ ∘ •■□
Key Evolving Cryptography	✓	✓		⋄⋆
Context Aware Transactions	✓	✓	✓	
Plenitude Rule	✓	✓	✓	⋄
Economic Finality	✓	✓	✓	∘
Trusted Execution Environment		✓		

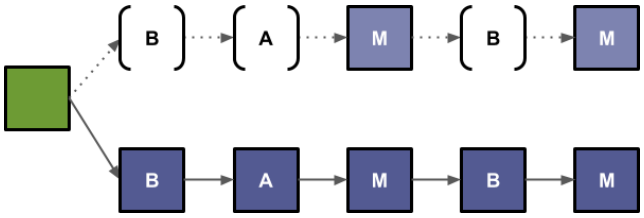

FIGURE 12. Blockchain snapshot up till point X. Validators have the same chance of producing blocks at any given branch. Even though the upper branch is Malory's fork, Bob and Alice are still elected as block validators even if they are not aware of that branch and may never produce a block for it (hence the forfeited blocks).

other branch does. This is because at a given slot one branch has at least 80% of generating a block while the other has at most 20%.

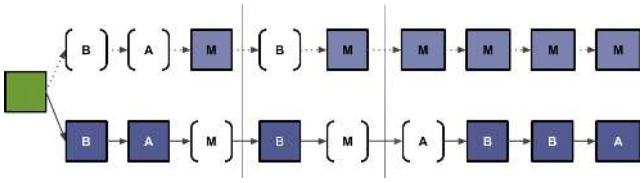

FIGURE 13. Blockchain snapshot where the density of the malicious branch is sparse for the first segments. While both branches have the same number of published blocks in total, if investigated, the first segments have very different block generation rates.

In the blockchain snapshot of Figure 13, we can observe how stake accumulation could work. The upper branch, which is the malicious one, in the first two segments, has a significantly sparser block generation in comparison with the third one. On the other branch, we can observe a smoother distribution in block generation throughout all segments.

Having empty blocks does not necessarily mean that a validator is blocking the chain on purpose, in this case, Alice. Sometimes, slots can be empty as no transactions were performed during a specific period. On the third segment, the upper chain has accumulated enough stake and block generation starts becoming denser. Once this happens, the adversary's stake will keep increasing in their own minted branch, contrary to the stake of the honest validators which will gradually decrease. As a result, blocks will be presented more often in the alternate branch and eventually overtake the main-chain. See Figures 14a and 14b.

The plenitude rule tries to determine the block density of a branch from its very formation, and use it as a measure to detect substantial changes to the block density. If we assume that the malicious validators will always be the minority[3] the main branch will always be denser than the competing branches. This rule enables us to identify whether an attack is taking place in a blockchain fork and identify the honest branch, defeating this way weak subjectivity.

F. ECONOMIC FINALITY

To disincentivise malicious behaviour by validator nodes on the blockchain, PoS implementations, such as Ethereum's PoS version [31], [34], employ mechanisms of *Slashing*, a financial punishment for identified misconduct of validators, which results in native blockchain currency loses. The idea is that if one or more validators leverage their stake in signing two different blocks at the same "height" (same point) in the chain, they risk losing their leveraged stake (their stake is "slashed").

With Economic Finality, the protocol ensures that all validators have something to lose once they misbehave, dealing this way a greater blow to the Nothing-at-Stake problem. Even though Economic Finality does not eliminate Long Range attacks, it does make them much less probable as validators are now further disincentivised in conducting any sort of attack against the network.

G. TRUSTED EXECUTION ENVIRONMENT

Storing data on a blockchain by its definition contradicts maintaining users' privacy. To address this problem several storage solutions like IPFS are proposed; however, secure multiparty computations (SMC) appear to be the only viable long-term solution. However, SMCs are currently far from efficient to adequately support arbitrary computations at the rate that modern blockchains require. Therefore, a promising alternative is the use of Trusted Execution Environments (TEEs) for securely executing part of the needed computations.

Software Guard Extensions (SGX) is the most prominent TEE technology today and available together with commodity CPUs for servers and desktops. SGX establishes trusted

[3]Note that in BFT based PoS protocols this minority has to be less than 33% of validators.

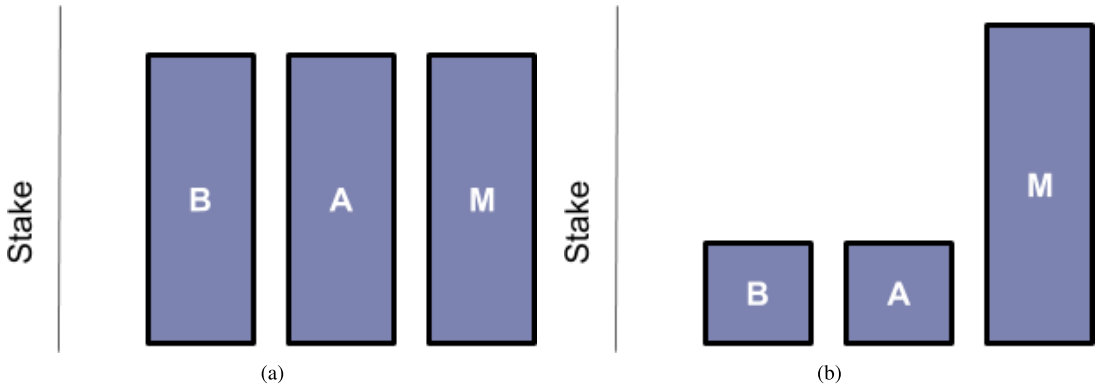

FIGURE 14. Stake distribution before and after the attack. After the attack Malory has the majority of the network stake. (a) Validator stakes at the genesis block. (b) Stake redistribution on Malory's branch after time X.

execution contexts, which are called enclaves, on a CPU, that isolate both data and programs from the underlying operating system, ensuring that the outputs are correct. An enclave might run only a small dedicated part of an application, or it may contain an entire legacy system, including some operating-system support.

Li *et al.* [45] propose the use of TEEs so that all signing can be performed within a trusted application and the signing keys are not available beyond the TEE. This way, the private keys are protected from being leaked to an adversary diminishing the chances of Long-Range attacks originating from hacking attempts. Essentially, this implies that all key pairs are generated from the TEE when a node first joins a blockchain. Apparently, a key issue in integrating such a solution is that each validator must also prove to its peers that she is using the TEE for all these procedures. The latter cannot be considered a trivial implementation issue hindering the adoption of this solution.

VI. CONCLUSIONS

Blockchain is a disruptive technology as it provides a decentralised architecture to store and process data among entities that do not trust each other. Therefore, it is being broadly adopted in various domains serving as the backbone of many applications [46].

Due to the fact that most of these applications are expected to have a big lifespan and in many cases important role, it is of utter importance to study their security and provide the necessary mechanisms to safeguard them. The great fragmentation in the field implies even more challenges as there are different protocols, implementations, and security measures that are needed in each case.

Long-range attacks are perhaps the biggest threat for PoS protocols as they are threatening one of the fundamental properties that people regard that blockchains have: immutability. A successful Long-Range attack will not simply change some blocks but would allow an adversary to completely rewrite the history of all transactions stored in a blockchain. As discussed, these attacks may not stem from the implementation of a specific protocol, but from its design, making it rather difficult to patch. Moreover, due to the nature of these attacks, the results maybe noticed only when it is too late.

In this work we analysed the reasons behind the existence of Long-Range attacks and how they can be performed. Moreover, we discussed the possible countermeasures reported in the literature and their efficacy. In Table 1 we present the different flavors of Long-Range attacks and the possible countermeasures. As discussed, none of these countermeasures can provide full protection from all these threats. Even more the proposed solutions are partial for each threat individually. Apart from timestamping and integrating the longest chain rule and moving checkpoints which seem to be integrated by all protocols, there is diversity in the integration of the rest of the countermeasures from the protocols. While the use of TEEs is very promising, they are not implemented by any of the protocols as their adoption implies hardware constraints.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Tech. Rep., 2008.
[2] V. Buterin and V. Griffith. (2017). "Casper the friendly finality gadget." [Online]. Available: https://arxiv.org/abs/1710.09437
[3] S. King and S. Nadal, "PPCoin: Peer-to-peer crypto-currency with proof-of-stake," Tech. Rep., Aug. 2012.
[4] P. Vasin. (2014). *Blackcoin's Proof-of-Stake Protocol v2*. [Online]. Available: https://blackcoin.co/blackcoin-pos-protocolv2-whitepaper.pdf
[5] Digiconomist. (2017). *Bitcoin Energy Consumption Index*. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption
[6] Malwarebytes. (2017). *Persistent Drive-by Cryptomining Coming to a Browser Near You*. [Online]. Available: https://blog.malwarebytes.com/cybercrime/2017/11/persistent-drive-by-cryptomining-coming-to-a-browser-near-you/
[7] N. Grech, M. Kong, A. Jurisevic, L. Brent, B. Scholz, and Y. Smaragdakis, "MadMax: Surviving out-of-gas conditions in Ethereum smart contracts," in *Proc. ACM Program. Lang.*, 2018, Art. no. 116.
[8] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia. (2017). "Dissecting Ponzi schemes on Ethereum: identification, analysis, and impact." [Online]. Available: https://arxiv.org/abs/1703.03779
[9] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on Ethereum smart contracts (SoK)," in *Proc. Int. Conf. Princ. Secur. Trust*. Berlin, Germany: Springer, 2017, pp. 164–186.
[10] I. Nikolic, A. Kolluri, I. Sergey, P. Saxena, and A. Hobor. (2018). "Finding the greedy, prodigal, and suicidal contracts at scale." [Online]. Available: https://arxiv.org/abs/1802.06038

[11] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3416–3452, 4th Quart., 2018.
[12] T. de Balthasar and J. Hernandez-Castro, "An analysis of bitcoin laundry services," in *Proc. Nordic Conf. Secure IT Syst.* Cham, Switzerland: Springer, 2017, pp. 297–312.
[13] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Gener. Comput. Syst.*, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17318332
[14] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the Internet of Things," *IEEE Access*, vol. 4, pp. 2292–2303, 2016.
[15] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 2017, pp. 357–388.
[16] M. Vukolić, "The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication," in *Proc. Int. Workshop Open Problems Netw. Secur.* Cham, Switzerland: Springer, 2015, pp. 112–125.
[17] A. Back *et al.*, "Hashcash—A denial of service counter-measure," Tech. Rep., 2002.
[18] M. Castro and B. Liskov, "Practical byzantine fault tolerance and proactive recovery," *ACM Trans. Comput. Syst.*, vol. 20, no. 4, pp. 398–461, 2002.
[19] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of BFT protocols," in *Proc. 2016 ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 31–42.
[20] Z. Zheng, S. Xie, H.-N. Dai, and H. Wang, "Blockchain challenges and opportunities: A survey," *Int. J. Web Grid Services*, vol. 14, no. 4, pp. 352–375, 2016.
[21] J. A. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol with chains of variable difficulty," in *Proc. 37th Annu. Int. Cryptol. Conf. Adv. Cryptol. (CRYPTO)*, Santa Barbara, CA, USA, Aug. 2017, pp. 291–323. [Online]. Available: https://doi.org/10.1007/978-3-319-63688-7_10
[22] W. Wang *et al.* (2018). "A survey on consensus mechanisms and mining strategy management in blockchain networks." [Online]. Available: https://arxiv.org/abs/1805.02707
[23] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.
[24] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99. 1999, pp. 173–186.
[25] J. Kwon, "TenderMint: Consensus without Mining," Draft v.0.6, Tech. Rep., 2014.
[26] V. Zamfir. (2017). *Casper the Friendly Ghost: A 'correct-by-construction' Blockchain Consensus Protocol*. [Online]. Available: https://github.com/ethereum/research/blob/master/papers/CasperTFG/CasperTFG.pdf
[27] J. R. Douceur, "The Sybil attack," in *Proc. Int. Workshop Peer-Peer Syst.* Berlin, Germany: Springer, 2002, pp. 251–260.
[28] B. N. Levine, C. Shields, and N. B. Margolin, "A survey of solutions to the Sybil attack," Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep., 2006, vol. 7, p. 224.
[29] G. O. Karame, E. Androulaki, and S. Capkun. (2012). *Two Bitcoins at the Price of One Doublespending Attacks on Fast Payments in Bitcoin*. [Online]. Available: http://eprint.iacr.org/2012/248.pdf
[30] J. Bonneau, "Why buy when you can rent? Bribery attacks on Bitcoin consensus," in *Proc. 3rd Workshop Bitcoin Blockchain Res. (BITCOIN)*, Feb. 2016. [Online]. Available: http://fc16.ifca.ai/bitcoin/papers/Bon16b.pdf
[31] V. Zamfir. (2015). *Introducing Casper 'The Friendly Ghost'*. [Online]. Available: https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost
[32] L. Xu *et al.*, "Enabling the sharing economy: Privacy respecting contract based on public blockchain," in *Proc. ACM Workshop Blockchain, Cryptocurrencies Contracts*, 2017, pp. 15–21.
[33] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Commun. ACM*, vol. 61, no. 7, pp. 95–102, 2018.
[34] V. Buterin. (2014). *Slasher: A Punitive Proof-of-Stake Algorithm*. [Online]. Available: https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm
[35] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, "Eclipse attacks on bitcoin's peer-to-peer network," in *Proc. USENIX Secur. Symp.*, 2015, pp. 129–144.
[36] P. Gaži, A. Kiayias, and A. Kiayias. *Stake-Bleeding Attacks on Proof-of-Stake Blockchains*. Accessed: Jan. 19, 2019. [Online]. Available: https://eprint.iacr.org/2018/248.pdf
[37] S. King and S. Nadal. *PPCoin: Peer-to-Peer Crypto-Currency With Proof-of-Stake*. Accessed: Jan. 19, 2019. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf
[38] (2014). *NXT Community*. [Online]. Available: http://wiki.nxtcrypto.org/wiki/Whitepaper:Nxt
[39] B. David, P. Gaži, A. Kiayias, and A. Russell, "Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Springer, 2018, pp. 66–98.
[40] G. Itkis and L. Reyzin, "Forward-secure signatures with optimal signing and verifying," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2001, pp. 332–354.
[41] M. Franklin, "A survey of key evolving cryptosystems," *Int. J. Secur. Netw.*, vol. 1, nos. 1–2, pp. 46–53, 2006.
[42] D. Larimer, "Transactions as proof-of-stake," Tech. Rep., Nov. 2013. [Online]. Available: http://www.wanglutech.com/LoadFiles/file_1483944101721_[%E7%99%BD%E7%9A%AE%E4%B9%A6]Daniel%20Larimer-%E5%B0%86%E4%BA%A4%E6%98%93%E4%BD%9C%E4%B8%BA%E6%9D%83%E7%9B%8A%E8%AF%81%E6%98%8E[%E8%8B%B1%E6%96%87][PDF%E4%B8%8B%E8%BD%BD].pdf
[43] J. Coleman, "Universal hash time," in *Proc. Ethereum Developer Conf.*, London, U.K., Nov. 2015. [Online]. Available: https://www.youtube.com/watch?v=phXohYF0xGo
[44] C. Badertscher, P. Gazi, A. Kiayias, A. Russell, and V. Zikas, "Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability," Tech. Rep., 2018.
[45] W. Li, S. Andreina, J.-M. Bohli, and G. Karame, "Securing proof-of-stake blockchain protocols," in *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Cham, Switzerland: Springer, 2017, pp. 297–315.
[46] F. Casino, T. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics Inform.*, vol. 36, pp. 55–81, Mar. 2018.

Authors' photograph and biography not available at the time of publication.