A security overview of Proof of Work protocol implementations under high transaction throughput

José Manuel Agüero Trejo

***Abstract*—Blockchain based systems have gained increasing popularity due to their public and decentralised approach to implementing cryptocurrencies, as well as their theoretical immutability with respect to transactions. Nonetheless, blockchain performance is slow compared to the transaction speed of established, high performance, conventional models. Scalability thus becomes a major concern to be addressed by blockchain implementations, however, naive optimisation of the transaction throughput exposes proof-of-work based blockchain systems to several vulnerabilities and attack strategies analysed in this paper.**

***Index Terms*—blockchain, GHOST, proof-of-work, liveness attack, balance attack, double spending.**

I. Introduction

Extensive analysis on the performance and security of proof-of-work systems has been motivated after Satoshi Nakamoto proposal [**?**] to incorporate a proof-of-work system as a key element in the development of the Bitcoin protocol. The work described by the proof-of-work protocol involves scanning for an specific value such that hashing yields a certain number of zero bits at the beginning of the output. Due to the complexity of the implemented hashing functions, such as SHA-256, the work required to obtain this value is exponential in the number of zero bits. On the other hand, exploiting the properties of a hash function, the verification is a trivial process.

This process can be implemented in a blockchain by designating a nonce and a target hash value that represents the block information. Hence, as blocks are added to the blockchain, the work required to modify an instance involves computing the proof-of-work of all the previously appended blocks. This property is intended to address several problems, such as preventing double spending and establishing scarcity in cryptographic currency protocols as well as determining representation in majority decision making (i,e. allowing a consensus to be reached).

Despite the advantages, there are important problems regarding energy expenditure, scalability, security and integrity of proof-of-work systems. In particular, the role of transaction processing, block generation and communication over a network play a crucial role in the security of such implementations.

As a consequence of scalability, higher block generation and transaction validation rates are desired; this, however, exposes the blockchain system implemented to several vulnerabilities. This paper aims to provide an analysis of the proof-of-work protocol implementations with respect to the transaction throughput.

II. Integrity in cryptocurrencies block chain systems

In a cryptocurrency system, transactions are managed and stored in a public ledger represented by the block chain; here, a transaction is confirmed to be legitimate by utilising a cryptographic signature. However ,when a central authority in charge of managing the scarcity of the currency and authenticity of transactions is not desired, preserving the amount of currency managed by the transaction mechanism poses a challenge.

This problem is addressed by utilising the proof-of-work protocol to collectively agree on the temporal order of transactions by trusting the longer chain, and therefore the transaction history with the most computational processing work associated. However, propagating a block over the network is not an immediate process; furthermore, due to the probabilistic nature of concurrency control, the possibility of concurrent additions of two or more independent blocks to the blockchain arises [**?**] generating two different valid block chains, this phenomenon is refer to as forking.

As a consequence, the possibility for a valid block that has been appended to the block chain to be removed at some point in time is present; *consensus finality* is therefore not satisfied [**?**]. Thus, in order to increase

the likelihood of transaction to remain on the validated ledger history several blocks need to be added to a fork and users have no immediate confirmation of transaction inclusion. Furthermore, a larger number of transactions is required to be processed as the system scales, however, as the transaction throughput increases, vulnerabilities commence to appear [**?**]; In particular, the system becomes increasingly exposed to attackers that attempt to replicate the conditions that enabled the broadcasting of another transaction on the network, this is referred to as the double spending problem.

III. Reduced security at high throughput exploited by double spending attacks

As mentioned before, the lack of *consensus finality* that characterises the proof-of-work protocol allows different forks to be generated as delays in the network occur. When several internally consistent valid block chains conflict, each node on the network needs to choose a ledger as the primary chain to follow and, eventually, all nodes must agree on the longer chain.

Although this forks can appear and be resolved naturally, forks can also be forced to occur with malicious intent by choosing to extend an arbitrary block. If the fork generated can produce a longer chain of blocks, the malicious transactions are accepted as valid. However, due to the large amount of computational work required to achieve this, such situation is assumed to be unfeasible by the proof-of-work protocol. Nonetheless, this is not an impossible feat. If an attacker has more computational processing power than the remaining nodes on the network, then generating blocks at a higher rate is possible. This is know as the 51% attack, and it plays an important role in exploiting the proof-of-work vulnerabilities. In particular, this idea is adopted to implement an attack based on the double spending problem.

In order to reverse transactions, an attacker may attempt to override the chain that the network has agreed on by creating a fork. By forcing a fork in the blockchain, the attacker may cause the ledger to erase or redirect a particular transaction; if the fork becomes long enough, it will eventually replace the main block chain and validate the malicious work intended to be performed. This is referred to as the *double spending problem*.

The difficulty of generating a fork is influenced directly by the block creation rate and the size of each block in the block chain, these parameters determine the transaction throughput of the protocol. Such values can be modified by manipulating the difficulty of the computational puzzle that miners are required to solve. The block generation rate is increased as the computational difficulty diminishes and blocks of larger size can be allowed to propagate on the network in order to increase the block size as desired. Nonetheless, scalability with careless manipulation of the aforementioned parameters leads to an increase in the potential number of forks generated from the main block chain. In addition, if an user is able to create blocks at a higher rate than the rate at which the main block chain grows, then the computational barrier that prevents attacks from being successful reduces its efficiency proportionally [**?**]. As a consequence, blocks are not necessarily added to the primary block chain, allowing an attacker to replace, modify or erase the target transaction information more easily; that is, to perform a *double spending attack*.

Variants of the Greedy Heaviest-Observed Sub-Tree (GHOST) policy (such as the one adopted by Ethereum) have been proposed to address this issue [**?**] [**?**]. The GHOST policy aims to ensure that the transaction history of the ledger converges to the same record, this is done by tracking the moment in which an individual block was either adopted or abandoned by all the participants on the network and using this information to assign a weight to each block in order to assign the fork with the largest weight as the main block chain; In other words, blocks that are not added to the blockchain still play a role in the chain selection process. The implementation of this policy increases the protection against 51% attacks when a high throughput rate is in action while being resilient to deficiencies due to network delays. Furthermore, it provides protection against Denial of Service attacks deployed by bursts of blocks that disrupt the network.

Two security properties become of crucial relevance in the analysis of the emerging vulnerabilities.

1) *common prefix*: this property defines the ability for honest participants two obtain the same view of the elements of the main public blockchain when a certain number of blocks is pruned before the current transaction validation interval.

2) *chain quality*: this property ensures that the chains with which honest participants are interacting does not contain a large number of conflicting blocks. At the same time, it allows transactions that have been processed a certain amount of blocks in the past, to persist in the main public blockchain.

This enables the definition of an important variable called *chain growth*, which expresses the rate at which a honest chain may grow while satisfying the previously established properties.

The possibility of attacks on proof-of-work systems integrating the variants of the GHOST policy remains a latent threat to the stability, scalability and integrity of such systems [**?**]. Two possible attacks that exploit the problem of transaction processing time will be discussed in the following sections.

A. Liveness Attacks

In [**?**], a class of attacks that exploits the transaction validation rate of the proof-of-work based blockchain implementations is introduced.

At a high level, the attacker attempts to delay the time it takes for a transaction to be reported at a block within a certain distance from the end of the ledger by an honest participant; we refer to this interval as confirmation time. Moreover, if a transaction is continuously provided to all the honest players on the network, eventually one of such players will include this transaction when generating a block; we denote this property as *Liveness*.

This class of attacks consists of three stages:

1) *Attack preparation:* An advantage is build by the attacker with respect to the honest participants. This can be done by adopting a selfish mining strategy [**?**] until the target transaction has been released.

   The idea behind selfish mining consists on constructing a pool of miners that share their capabilities to perform computational work over a network to increase their probabilities of building a new block. As the pool obtains new blocks at a higher rate than the rate achieved by honest participants, they are able to develop a longer private block chain.

   Eventually, as the public primary blockchain approaches the malicious chain with respect to the number of blocks, the selfish miners reveal their own private chain to force a change due to the nature of the proof-of-work protocol. This leads to an unfruitful resource expenditure by the honest participants and encourages other players to join the selfish mining pool.

2) *Transaction denial:* With the aim of delaying the target transaction from being processed, the attacker attempts to prevent blocks generated by honest participants that contain such transaction from being added to the public blockchain.

   This is done to extend, as much as possible, the selfish blockchain; only blocks without the target transaction are added to the private chain and broadcasting occurs selectively when honest participants catch up.
   At this point, the attackers continue working on the chain that does not contain the target transaction, instead of working on the longest chain. When the selfish chain can no longer provide an advantage over the public chain and, as a consequence, the inclusion of the target transaction can no longer be delayed, the attackers proceed to the next stage.

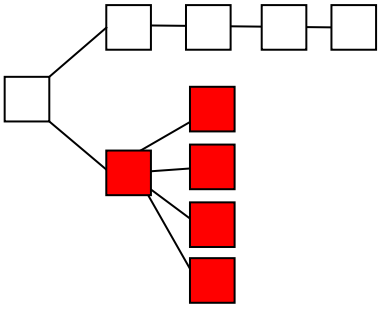
3) *Blockchain retarder:* The attacker attempts to decrease the rate at which the blockchain that includes the target transaction grows. As transaction validation only occurs after a certain number of blocks have been added since the target transaction was included, successfully decreasing the chain growth rate effectively delays the transaction confirmation time. Here the regular bitcoin implementation and improved GHOST variants are addressed in a different manner.

   In the bitcoin implementation, the attacker can simply stay in the second stage until further delay is impossible. However, systems deploying some variant of the GHOST policy require more work.

   The GHOST policy [**?**] assigns a weight to every block that is generated; thus, when a block is added to a chain, the total weight of the sub-chain is characterised by the sum of the weight of all blocks included. Hence, when the addition of blocks lead to forking, the

sub-chain with the largest weight is designated as the main blockchain. However, a participant is not prevented from continuing to add blocks to a particular sub-chain. Instead, since every block (regardless of its inclusion in the main blockchain) contributes to the overall weight of the main blockchain, the policy simply aims to deter an attacker from overruling the consensus by simply generating a longer chain.

In order to exploit this policy, the attacker tries to generate many different blocks pointing to the same preceding block in a particular sub-chain to increase its weight. The following diagram illustrates this process where the red squares represent the malicious blocks.



If the honest blockchain gets heavier, the attacker can broadcast the malicious shorter chain to overrule the honest participants consensus. By continuously repeating this process, the rate at which the blockchain grows is effectively reduced, therefore delaying the target transaction validation time.

B. Balance attacks

This form of attack exploits the GHOST policy implementation by taking advantage of the transaction processing time, this is done by delaying the rate at which interactions between multiple subgroups of participants occur over a network where a balanced computational processing power is assumed. This is based on the idea that block propagation delay can lead to wasteful block production reducing the amount of computational power required to perform an attack [**?**].

The attack consists of issuing transactions in a particular subgroup and generating blocks in a different subgroup in order to produce a heavier sub-chain that overrules the chain that includes the target transactions; methods such as the one described in the Liveness attacks can be implemented for this purpose.

The relevance of this kind of attacks lies in the ability for an attacker to work on an isolated sub-chain before merging into the public process to directly influence the chain selection policy. Thus, balance attacks allow a violation of the main blockchain integrity, thus making it vulnerable to double spending by rewriting previously validated transactions. Although disrupting communication between subgroups is a vital and difficult part of this attack's implementation, such interference has been shown to be successful in the past [**?**] [**?**] [**?**].

To execute this attack, information about the logical and physical communication system, computational processing power of participants and the difficulty of the cryptographic challenge must be known. Some of this data is public and the network information can be obtained by exploiting the physical or logical overlay of peer-to-peer networks in which the blockchain protocol is implemented.

Balance attacks therefore highlight the trade-off between communication delays, computational processing capabilities and the cryptographic puzzle difficulty. It follows that a mining pool may obtain the potential for double spending when the computational processing power possessed is high and the transaction throughput allows for exploitable delays, which is particularly susceptible to the complexity of the cryptographic challenge, as the resolution of the puzzle influences the block generation rate.

This important observations allow for further work in the analysis of systems that implement proof-of-work protocols as this attack setting is highly reliant of easily collected participant statistics; the possibility for extending these results to settings in which the computational processing power of the participants involved is not readily available remains. Furthermore, the possibility of implementing man-in-the-middle attacks and Denial of Service attacks to introduce delays on the network remains to be a relatively unexplored area.

C. Majority attacks

The vulnerabilities previously discussed motivates the development of evaluation frameworks in order to detect and devise strategies to addresses the problems encountered with proof-of-work protocol

implementations [**?**] [**?**]. However, the essential problem of preventing majority attacks persist in consortium based blockchain networks as well as playing an important role in the previous attack strategies that clearly benefit from the majority based vulnerabilities. Different approaches have been devised for this purpose.

The introduction of a second cryptographic challenge to the proof-of-work protocol has ben presented in [**?**] to prevent large public mining pools to be formed.

The proposed strategy works as follows: In addition to the block header hash cryptographic challenge, characteristic of a traditional proof-of-work protocols, a new target set by the network must be met. Here, the private key that controls the coin-base payout address from the participants is required for signing the header in order to complete the hashing procedure. This effectively forces the mining pool operators to distribute their private key in order to retain decentralisation. However, this requirement deters participants from constructing large mining pools, as the private key distribution allows the mining pool clients to authorise the manipulation of funds from the payout address for their own interests. This forces mining pools to privatise their mining equipment in order to continue working in a similar manner. Thus, public mining pools are forced to reduce their size as a consequence. It is important to emphasise that the rate of miner equipment development becomes an important factor in the performance of the discussed model.

Such paradigm allows for the potential to extend the previous analysis with the implementation of the proposed double challenge strategy.

IV. Conclusion

In this paper we provided a strong insight into the security challenges of scaling proof-of-work based blockchain systems due to the direct influence of transaction throughput on the security of the protocol implementations. Different attacks strategies were explored emphasising the reoccurring problem of double spending when communication delays affect the system implementation as a consequences of malicious parties. The role of blockchain generation and chain growth rate is highlighted; similarly an analysis of the computational processing challenge influence on the security transaction trade-off is provided. Finally, an analysis of the role of mining pools and possible alternatives to deter their creation is explored. Thus, a comprehensive overview of the challenges and vulnerabilities that arise as a consequence of high transaction throughput on proof-of-work based blockchain systems provides the groundwork for future work involving evaluation and prevention frameworks.

A. Exam question proposal

Finally, the following question encompassing the content of this paper and the class presentation is proposed:

As previously discussed, proof-of-work based blockchain implementations are prone to the phenomenon called forking. How can the proof-of-work protocol be exploited by a malicious agent through the use forking?