Bitcoin Revisited: Formalization, Benchmarking, and Open Security Issues

SUBMISSION DATE / POSTED DATE

16-08-2022 / 23-08-2022

CITATION

Baniata, Hamza; Kertesz, Attila (2022): Bitcoin Revisited: Formalization, Benchmarking, and Open Security Issues. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.20496492.v1

DOI

10.36227/techrxiv.20496492.v1

Bitcoin Revisited: Formalization, Benchmarking, and Open Security Issues

Hamza Baniata[a], Attila Kertesz[a]

[a]*Department of Software Engineering, University of Szeged, Szeged, 6720, Hungary*

Abstract

Thousands of researchers and practitioners around the world neglect foundational understanding of the most trusted cryptocurrency platform, Bitcoin. According to the typical tendency of taking the discussion of Bitcoin's security for granted, it is usually confidently stated that, technically, Bitcoin is one of the most secure platforms for investment. In this paper, we revisit the fundamental calculations and assumptions of this widely adopted platform. We break down the original calculations in order to better understand the underlying assumptions of the Bitcoin proposal. Accordingly, we set open research questions, and we highlight expected scenarios upon the violation of each assumption. Furthermore, we propose a novel formalization of the Bitcoin mining problem using the Birthday Paradox, which we utilize to propose new concepts for benchmarking the security of Bitcoin, including the Critical Difficulty and Critical Difficulty per given portion. Our calculations indicated that it would be profitable to launch Partial Pre-Image attacks on Bitcoin once the mining puzzle difficulty reaches 56 leading zeros. Additionally, we discuss how Quantum Computing can be used to attack Bitcoin, and the implications of Sharding on the security of Bitcoin. The main objective of this work is to highlight, demystify, and justify the confusion between the realistic relative trust versus the common unconditional trust in Bitcoin.

Keywords: Blockchain, Proof-of-Work, Bitcoin, Security, Hash Functions

1. Introduction

Bitcoin [1] is the first solution that addressed several open issues of a wanting online distributed cryptocurrency system. The most important aspects that delayed the proposal of such a reliable system are the security, privacy, and full-decentralization. That is, the success of a given online distributed cryptocurrency solution is attributed by the reliable provision of high security and privacy measures, without using a Trusted Third Party (TTP). Several techniques were deployed within the proposal of Bitcoin to comprehensively address those issues, including Proofs-of-Work [2], robust hashing functions [3], Merkle Trees [4], and distributed timestamps [5]. The combination of those techniques resulted in the emergence of the so-called Blockchain technology.

The requirements set out in Nakamoto's paper have been reliably proven, both theoretically and experimentally. As we can see, Bitcoin has been increasingly used and adopted for more than 13 years now (as of June 2022, by the time of writing). Not only Bitcoin, but the Blockchain technology in its generality was also used in a wide variety of applications, including distributed e-voting [6], e-Health [7], IoT [8], smart contracts [9] and IoV [10].

Upon the proposal of Bitcoin, several main security-related assumptions were made depending on the postulated high security of the utilized cryptography methods. As long as these main assumptions were not violated, the claimed high security of Bitcoin shall remain dependable.

In this paper, we discuss and simplify those main security-related assumptions, and we study when and how those assumptions could be violated. Specifically, we discuss probable alternative Bitcoin mining methods, other than those presented in the original paper [1]. To do so, we present a novel formalization of the Bitcoin mining problem using the Birthday Paradox Problem [11]. Additionally, we dive into the intriguing challenge of Quantum Computing [12] methods against Bitcoin's security, and we discuss how and when, such methods could violate the basic Bitcoin assumptions. Finally, we discuss the applicability of state-of-the-art Sharding [13] approaches, and we set main requirements of a secure Bitcoin sharding protocol. Throughout the paper, we present several related open issues, and we systematically discuss novel, and previously proposed, Bitcoin security benchmarks.

The remainder of this paper is structured as follows: Section 2 provides necessary background of Blockchain, SHA-256, and Bitcoin mining. Section 3 provides detailed discussion on the major assumptions of Bitcoin and clarifies its equations. Section 4 presents our proposed formalization model of Bitcoin mining using the Birthday Paradox. Section 5 delves into attack approaches that violate Bitcoin assumptions, including an alternative Brute-force attack, Partial Pre-image attack, Quantum attacks, and Blockchain Sharding. Finally, Section 6 concludes our work. To facilitate our discussions in the remaining part of our paper, we list the main notations and abbreviations used in Table 1.

2. Background

2.1. Blockchain

A BC-based system is characterized by its infrastructure, data structures, networking model, and Consensus Algorithm

Table 1: Descriptions of notations and abbreviations used throughout the manuscript

Notation	Description
V	Set of nodes in network $G(V, \epsilon, w)$
ϵ	Set of edges in network G
$e_{i,j}$	Edge $e \in \epsilon$ connecting nodes $i, j \in V$
$w_{i,j}$	Weight on $e_{i,j}$
N	Number of nodes in the set V
M	Set of honest nodes in V
K	Set of faulty/adversarial nodes in V
f	Number of nodes in the set K
T	Total computational power in BC
a_i	Computational capacity of $i \in V$
q	Portion of computational power controlled by K out of T
p	Portion of computational power controlled by M out of T
Ψ	Tolerated upper bound fraction of f out of N, or of q out of T
$h(.)$	Hashing function
Φ	Probability the next valid block is found by a miner $\in V$
E	Probability $m_i \in M$ solves PoW puzzle
Q	Probability $k_j \in K$ solves PoW puzzle
Ω	Non-Zero digits allowed for a correct PoW
r	Zero digits required for a correct PoW
Abbreviation	Description
BC	Blockchain
DL	Distributed Ledger
CA	Consensus Algorithm
TX	Transaction
B_z	Most recent confirmed Block
B_{z+1}	Next Block
B_{z-u}	Block at depth u
P2P	Peer-to-Peer
PoW	Proof-of-Work
UTXO	Unspent TX Output
Nonce	Number-used-Once

(CA). The infrastructure can be formally described by a set of N nodes $V = \{v_1, v_2,..v_N\}$, usually termed as miners. Data shared between elements of the set V is described according to the application of the system. For example, transactions (TXs) are submitted by end users to the BC network so that they are processed and added to its Distributed Ledger (DL). Usually, TXs are shared with all miners triggering them to generate new blocks of data. A block usually consists of a header and a body. The header may consist of data such as the type of block, the type of CA, the timestamp, the hash of the body and, most importantly, the proof of block validity. The body, on the other hand, usually includes a group of TXs and the hash of the previous block body. More technical details can be found in [14].

As BC nodes form a distributed system, those nodes exchange data through a P2P network and communicate by message passing via directly connected links. BC nodes connect to their peers once they are granted access to the network, making them demonstrable as a graph $G = (V, \epsilon, w)$ of a connected giant component, where ϵ is the set of edges in G, representing the communication lines between the elements of V. Each $e_{i,j} \in \epsilon$, connects exactly two nodes $i, j \in V$, and can be traveled in both directions. Each $e \in \epsilon$ is associated with a distinct non-negative value, namely weight ($w_{i,j}$ or w_e), which represents the transmission time needed to deliver 1 bit of data from node i to node j or vice versa, computed in ms. A sub-graph of G is any graph $G' = (V', \epsilon', w')$, such that $V' \subseteq V$ and $\epsilon' \subseteq \epsilon$. G' is also undirected and weighted as it inherits the properties of the original graph, yet it is not necessarily connected.

Let $K = k_1, k_2, ..., k_f$ be the set of adversary nodes in G and $M = j_1, j_2, ..., j_{N-f}$ be the set of honest nodes, Equation 1 is usually assumed valid for BC networks.

$$M + K = V \tag{1}$$

Let the network's total computational power be T. The attacker, then, controls a portion q out of T that can be calculated using Equation 2, where a_i is the computational capacity associated with node $i \in K$.

$$q = \sum_{i=1}^{f} \frac{a_i}{T} \tag{2}$$

Let the remaining portion p out of T be controlled by nodes

in the set M. p can then be calculated as in Equation 3, where a_j is the computational power of $j \in M$.

$$p = \sum_{j=1}^{N-f} \frac{a_j}{T} \tag{3}$$

Every BC-based system must operate a CA in order to maintain the consistency of its DL [15]. As tens of CAs were proposed in the literature, a CA is usually considered *valid*, if it was proven secure under specific formalized circumstances. One of the main benchmarks used to describe the security level of a given CA is its tolerance for adversary nodes out of the total number of nodes, denoted as Ψ. The tolerance benchmark Ψ is typically mathematically represented by an inequality that relates either q with T or f with N. As long as Ψ holds, the BC is considered secure.

For example, the Delegated Byzantine Fault Tolerance (dBFT) algorithm [16] was proven secure as long as $f <= \frac{N-1}{3}$, while the most famous CA used in Bitcoin, known as Proof-of-Work (PoW) algorithm [1], was proven secure as long as Condition 4 holds.

$$q < T/2 \tag{4}$$

2.2. SHA-256

A hashing function, or a one-way encryption function, $h(.)$ is a mathematical function that takes a variable-length input string and converts it into a fixed-length binary sequence that is computationally difficult to invert [17]. A hashing function enables the determination of a message's integrity: any change to the message will, with a very high probability, result in a different message digest[18].

As an encryption method, hashing functions have been studied and improved over the years to guarantee the highest possible security. Main categories of attacks on hashing functions can be classified as follows:

1. **Collision attack** [19]: This attack tries to find two inputs producing the same hash value, i.e. a hash collision. Classically, a collision attack is described as follows: Find *any* two different messages $m1$ and $m2$ such that $h(m1) == h(m2)$.
2. **Preimage attack** [20]: This attack tries to find a message that has a specific hash value. That is, given only the hash value $h(m)$, a pre-image attacker attempts to recover *any* m' such that $h(m') == h(m)$.
3. **Second-Preimage attack** [21, 22]: Given an input $m1$, try to find another input, $m2$ (not equal to $m1$) such that $h(m1) == h(m2)$.
4. **Length extension attack** [23]: In this attack, the attacker can use an available $h(m)$ and the length of m to calculate $h(m||m')$, where m' is an extension forged by the attacker. This attack is successful if the attacker could run it without needing to know the content of m.
5. **Brute-force attack** [24]: Simply put, this attack implies sequential, or random, testing of a wide range of inputs, until finding the correct or desired output. Performing such attack on hashes is considered the easiest to implement but the hardest in terms of cost. There are several ways to use such attack as it can be used to run any of the previously described attacks. For example, if an attacker needs to know a message m that was used to output the hash $h(m)$, the attacker may sequentially try all possible inputs, hash each input, and check each hash of each input if it is equal to the desired $h(m)$. Once an input m' is found where $h(m') == h(m)$, the attacker may provably claim that $m == m'$.

Accordingly, the five main properties of a powerful hashing function [25] are:

1. **Fixed size of output**: the function takes variable length input and always outputs a string with the same predefined length.
2. **Preimage resistant**: given the output, it should be mathematically inefficient to reverse-engineer the original input.
3. **Second Preimage resistant**: given the input and output, it should be mathematically inefficient to obtain a second input that produces the same output.
4. **Collision resistant**: it is computationally infeasible to find any two inputs that produces the same output.
5. **Random distribution of outputs**: If any single bit of the input is changed, the function will produce an entirely different output.

SHA-256 [3] was proven secure against all known attacks on hashing functions, except for, trivially, the Brute-force attack [26, 27]. Specifically, the lower bound complexity of algorithmic Preimage or Second-Preimage attacks on SHA-256 was evidently reported to be 2^{256}, while the lower bound complexity of algorithmic Collision attacks was evidently reported to be 2^{128} [28]. As such bounds make the SHA-256 compliant with all above mentioned properties, it has been deployed in Bitcoin as the main utilized hashing function.

2.3. Bitcoin Mining

The Bitcoin [1] mining problem is defined as finding a nonce which, together with a given block of data m, produces $h(m)$ that complies with the puzzle difficulty. The puzzle difficulty in Bitcoin is defined as the dynamically pre-defined number of leading Zeros r in the produced hash. This produced hash, together with the block of data is called a valid block, while the nonce that could produce the hash is called the Proof-of-Work (PoW).

Remark 1. *We assume throughout this paper that there is only one required nonce to search for. The effect of using more than one nonce [29] may need further analysis and modifications.*

As it is thus far argued that the only way to find such a PoW is by conducting a Brute-force attack, miners should be confident a miner that provides a correct PoW, has worked for a sufficient time prior to proposing the valid block. This mining

time window should be sufficient for data to propagate throughout the network before the next valid block is produced by another miner. Accordingly, the consistency of local views of confirmed blocks, at different physical locations of the network, is maintained. To realize this, the puzzle difficulty is regularly modified by the miners referring to the average time between consequent blocks confirmed within the preceding two weeks, and a hard-coded time window in the Bitcoin's protocol (i.e. 10 minutes). Miners that adhere to the rules of the above description are called honest miners, and are called adversary miners otherwise.

3. Simplifying Bitcoin's Assumptions and Calculations

Lets break down the original *Calculations* of Bitcoin into simpler pieces. In the original paper, only one attack model was analyzed, where an attacker controls q. Assuming Equation 1 holds, Equation 5 was adopted.

$$p + q = 1 \tag{5}$$

Abstractly, the main objective of the attacker is to find the next block, containing its fraud TX, including a valid PoW, before one of the honest nodes in the network finds a valid, non-fraud block. In this model, the two portions of the network, i.e. attacker and honest portions, are racing towards finding the valid block.

The Bitcoin proposal was mainly built upon the following three major assumptions:

1. **Attack and honest mining mechanism is unified**: Both types of miners use the same mechanism to find a PoW for any given block (i.e. that does or does not include a fraud TX),
2. **Attack and honest resources requirement is unified**: Generating a valid PoW using such mechanism requires only computational power, and
3. **Honest majority**: At any given moment, $p > q$ is valid. Accordingly, honest miners are expected to have higher probability to find the next valid block.

Following these assumptions, it was assumed that the only factor affecting the probability of an attack is p. Because of that, portion value notations (i.e. p and q) were also used to denote the probabilities of successful mining by both M and K, respectively. To clarify this mathematically, let Φ denote the probability the next valid block is found by any given miner $\in V$, and $\overline{\Phi}$ be the probability the next valid block is NOT found by that miner, then:

$$\Phi + \overline{\Phi} = 1 \tag{6}$$

The probability a miner $\in M$ finds the next block, denoted as E, should then be calculated using Equation 7, while the probability a miner $\in K$ finds the next block, denoted as Q, should be calculated using Equation 8.

$$E = p \times \Phi \tag{7}$$

$$Q = q \times \Phi \tag{8}$$

Although it was not mentioned clearly, Bitcoin calculations assumed that $\overline{\Phi}$ is always equal to 0, which resulted in Equation 9 (by summing Equations 7 and 8, with reference to Equation 5, and depending on the correctness of the first two assumptions). This equation justifies why p and q were used instead of E and Q, respectively. This special case was evaluated in the original paper, using Equation 10, to compute the attacker's potential progress λ for a given block at depth z. Several other inaccuracies of the formalization and validation in the original Bitcoin proposal were discussed in [30].

$$E + Q = 1 \tag{9}$$

$$\lambda = z\frac{q}{p} \tag{10}$$

By the time of writing the original paper, there was no evident algorithm proving an efficient method to run Preimage attacks on SHA-256. However, several researchers [20, 31, 32, 33, 34, 35] attempted, and still attempting, to run such attacks. As long as a preimage attack cannot be run on SHA-256, the *only* approach available to solve Bitcoin's mining problem is to run a Brute-force attack (first assumption), which only requires computational capacity (second assumption).

The third assumption cannot be anyhow controlled or guaranteed by any system entity as there is no TTP. However, it was argued in the original paper that even if this assumption does not hold anymore, it would be more profitable for an attacker to play by the rules. That is, use their controlled portion of computational power to create value for themselves (mint out of thin air), rather than undermine the system and the validity of their own wealth. This argument does not seem to be scientifically convincing as justifications for undermining the system may vary. Nevertheless, if the attacker indeed followed this argument, it is easily discoverable whether the third assumption does not hold anymore (i.e. using tools such as [36]). Even with an attacker that follows the rules, the system would lose its credibility against user trust, once it was found that $q > p$.

The remainder of this work attempts to answer the following main research questions:

RQ 1. *What are the cases in which Equation 5 cannot be substituted by Equation 9?*

RQ 2. *What are the cases in which one, or both, of the first two assumptions do not hold?*

RQ 3. *What are the implications of violating Bicoin's assumptions?*

4. Formalizing the Bitcoin mining problem using the Birthday paradox

We can formalize the Bitcoin mining problem using the Birthday Paradox [11] as follows. Let the set $H = 1, 2, ..., \xi_H$ consist of all probable numbers that, if concatenated with r Zeros, would produce a correct hash. A correct hash h is then

defined as $h = (0 \times r) + hc$ where $hc \in H$. For simplifying our description, let the set H' consist of all correct hashes, and the set H'' consist of all incorrect hashes. The relation between r and Ω is defined in Equation 11.

$$r = 64 - \Omega \tag{11}$$

Remark 2. *The number of elements ξ in H (and H', denoted ξ_H and $\xi_{H'}$, respectively) is trivially $2^{4\Omega}$, since the hash is decoded at base 16. ξ_H and $\xi_{H'}$ are bounded for SHA-256 ($1 \leq 2^{4\Omega} \leq 2^{256}$).*

Remark 3. *The number of elements in H'', denoted $\xi_{H''}$, is $2^{256} - 2^{4\Omega}$.*

Furthermore, let the set $C = 1, 2, ..., \xi_C$ consist of all values that a miner would test as a nonce, for a given block, before it drops the mining task of this block. The number of elements in C (ξ_C) is equal to the last nonce a miner would test[1]. As of March, 2022[2], the default last tested nonce in Bitcoin is 2^{32} [37]. Now, let the set HC, with length ξ_{HC}, be a merged set of H and C. ξ_{HC} can then be defined using Equation 12.

$$\xi_{HC} = \max\{\xi_C, \xi_H\} \tag{12}$$

Let $test(hc_1, hc_2) \rightarrow \{0, 1\}$ be a function that tests pairs of elements $(hc_1, hc_2), \forall hc \in HC$, where its output is 1, if mining hc_1 would produce $h = (0 \times r) + hc_2$ (or vice versa), and is 0 otherwise. Let MATCH be the event where $test(hc_1, hc_2) \rightarrow 1$, while NON-MATCH be the event where $test(hc_1, hc_2) \rightarrow 0$. We can also define the potential of using the set C to produce a MATCH using Equation 13.

$$\varpi = \frac{\xi_C}{\xi_{HC}} \tag{13}$$

The MATCH probability for any pair of elements in the set HC, equals 1 divided by the combined lengths of the sets H' and H'' (i.e. $\xi_{H'} + \xi_{H''} = 2^{256}$ as per Equation 11 and Remarks 2 and 3). For any two pairs, it would be $2/(\xi_{H'} + \xi_{H''})$. Consequently, the probability of a MATCH Φ, and the probability of Non-Match $\overline{\Phi}$ can be calculated using Equations 14 and 6, respectively.

$$\Phi = \frac{\xi_{HC}}{\xi_{H'} + \xi_{H''}} \tag{14}$$

However, $\overline{\Phi}$ can be calculated, according to the Birthday paradox, using Equation 15, which results in Φ being calculated using Equation 16. Figure 1 represents the approximate MATCH and NON-MATCH probability distribution for sampling $\xi_{HC} \in [0, 100]$ and $\xi_{H'} + \xi_{H''} = 100$) using all these equations.

$$\overline{\Phi'} = \prod_{y=1}^{\xi_{HC}-1} \frac{(\xi_{H'} + \xi_{H''}) - y}{(\xi_{H'} + \xi_{H''})} \tag{15}$$

$$\Phi' = 1 - \overline{\Phi'} \tag{16}$$

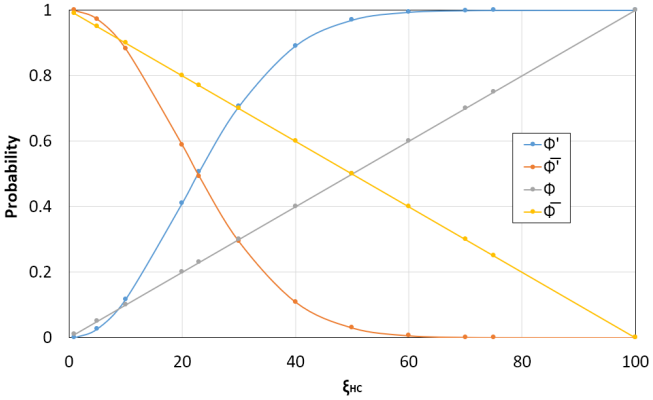


Figure 1: Graphs showing the approximate MATCH and NON-MATCH probability for $\xi_{HC} \in [0, 100]$ and $\xi_{H'} + \xi_{H''} = 100$, using Equation 14 (Gray), Equation 6 (Yellow), Equation 15 (Orange) and Equation 16 (Blue)

Naturally, both Equations 14 and 16 can be deployed in Equations 7 and 8, which can be used for both honest and attacker miners, respectively, as long as the first two assumptions hold. Putting it differently, if both honest and attacker parties would have equal MATCH probabilities, Equation 9 would still hold. However, in the case where one, or both, of the first two assumptions were violated, Q would require further considerations depending on the attack model (thus Equations 8 and 9 may not be valid).

5. Implications of Violating the Bitcoin's Assumptions

In this section, we present several attack approaches on Bitcoin. The first approach is a Random Brute-force attack exemplifying an attack that only violates the first assumption. The second approach is a Partial Pre-image attack exemplifying an attack that violates both, the first and the second assumptions. The third approach is covered using Quantum Computing, giving examples of attacks that is assumed to only violate the second assumption. However, we comprehensively show that the third assumption is also required to be violated.

Implications of violating the third assumption alone have been already discussed previously in the paper, and they can be concluded by an attacker taking over the system. Regardless of the attacker intentions of maintaining an honest or adversary behaviour, such violation shall drop the credibility of, and the trust in the system. We finalize our work by discussing realized cases, where the third assumption can be violated in sharded Bitcoin.

5.1. Alternative Brute-force puzzle solution

As honest miners search sequentially for a nonce that fulfils the puzzle difficulty, we propose that the attacker searches randomly for such a nonce. Assuming the outputs of the hash func-

[1] 4294967295 (0xffffffff): docs.microsoft.com/en-us/cpp/c-language/cpp-integer-limits?view=msvc-170

[2] github.com/bitcoin/bitcoin/blob/640eb772e55671c5dab29843cebe42ec35cb703f/src/rpc/mining.cpp

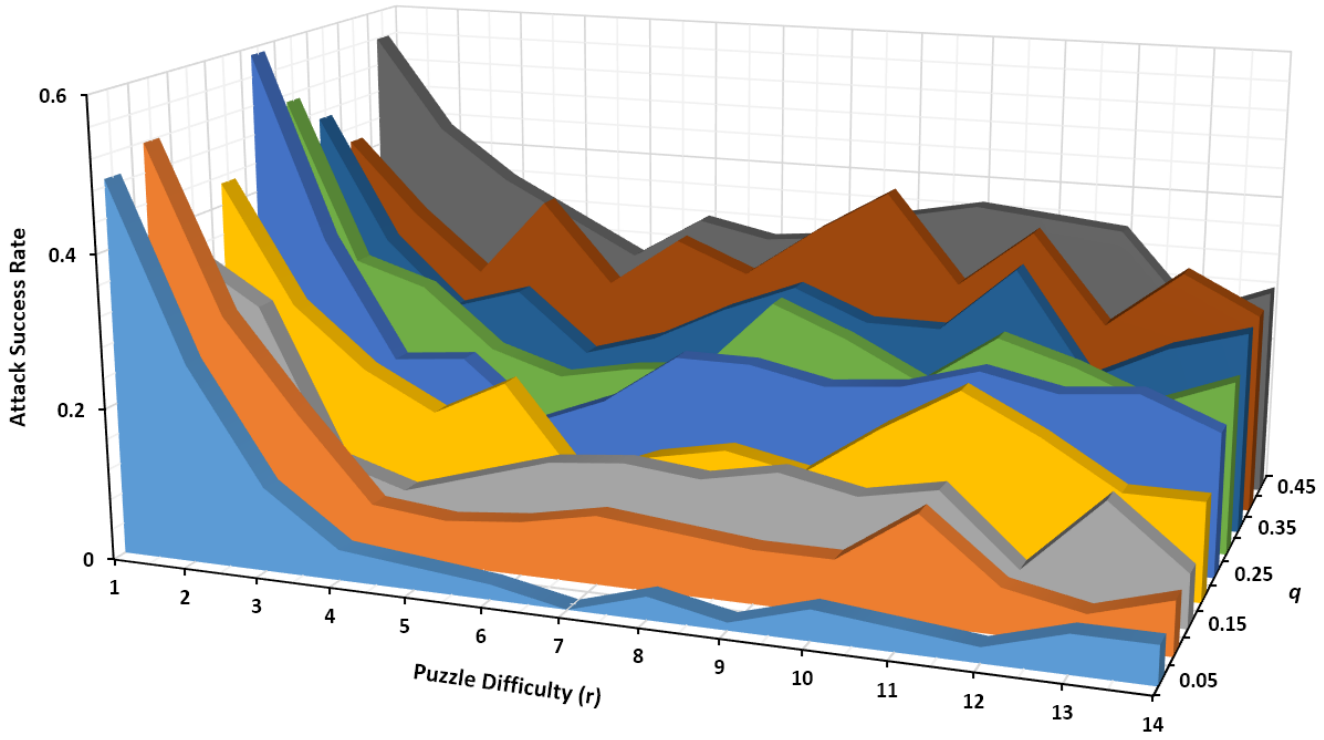


Figure 2: Attack success rates when using Algorithm 1 with different parameterization of puzzle difficulty and portion controlled by the attacker (q)

tion and the inputs of the attacker were *equivalently* Truly Random [38], an adversary miner should theoretically have higher chances to find a puzzle solution before an honest miner does. Note that both the attacker and honest miners are deploying the Brute-force mechanism to find the solution and, thus, the second assumption is not violated. However, the first assumption is violated as the mechanism is deployed differently.

The original Bitcoin calculations primarily rely on p and q values for determining the probability of a successful attack (on a block at depth z). That is, hardening the puzzle is assumed to have no effect on such probability depending on the correctness of the first two assumptions. In other words, it is assumed that hardening the puzzle difficulty would equally affect Q and E and, thus, Equation 9 should always hold regardless of the value Ω.

We propose Algorithm 1, representing a dummy attack procedure, to check if this assumption is correct. In the proposed algorithm, the attacker prepares TX_2 that consists incorrect data. The attacker then waits for a new block (B_z) to arrive. Once B_z is received, a transaction within (say TX_1) is modified so that TX_2 becomes correct. The Collatz_Update function exploits the Collatz Conjecture [39, 40] to guarantee a Truly Random search for a correct $nonce_A$. An attacker that uses our proposed algorithm might have an advantage over honest miners, in finding a correct nonce if, and only if, the following conditions were fulfilled:

1. The first value of $nonce_A$ was accurately selected for both B_z and B_{z+1}, and
2. The randomness uniformity of the Collatz Conjecture maps to the randomness uniformity of SHA-256 output range (i.e. relatively more than that of a sequential search).

More sophisticated randomization approaches using, specifically, the Collatz Conjecture were proposed in the literature. For example, several pseudo-random number generators that could pass the NIST Test Suite were proposed in [41]. Out of these generators, one was proven to generate uniformly distributed output. On the other hand, the randomness margin of the SHA-256 hashing function was reported to be 0.734 in [42], which raises Research Question 4, whose answer shall be the perfect candidate to substitute our randomization approach.

RQ 4. *What is the randomization approach, whose outputs best map to the SHA-256 randomness uniformity?*

We implemented Algorithm 1 using Python3, on top of which we built a simulator that consecutively generates 1,000 Bitcoin blocks for each given pair of unique parameterization (r, q). We tested all $r \in [0, 1, ..., 14]$ and all $q \in [0, 0.1, ..., 0.45]$, resulting in 675 pairs and a total of 675,000 attempts to attack the simulated Bitcoin Blockchain. Accordingly, we calculated the experimental attack success rate using Algorithm 1 by dividing the number of successful attempts by the total number of attack attempts, per unique pair. We ran our code on the Google Cloud Platform using a VM of type c2-standard-4 (4 Intel Cascade Lake vCPUs clocked at 3.8GHz with 16GB of RAM), running Ubuntu 18.04 LTS. Our code is publicly available at Github[3]. Part of the results of our experiments are pre-

[3]

Algorithm 1: Dummy Brute-force Attack on Bitcoin's puzzle

Input:
B_z = Most recently confirmed valid block;
Output: B_{z+1} = Forged valid block;
Function *Collatz_Update (nonce$_A$, tried_ONE_already)*
    if *nonce$_A$ is even* **then**
        nonce$_A$ = *nonce$_A$* / 2
    else
        nonce$_A$ = $(3 \times nonce_A) + 1$
    end
    if *nonce$_A$ == 1 and tried_ONE_already* **then**
        nonce$_A$ = random.randint(5, maximum_int)
    end
return *nonce$_A$*
Function *Forge_block (TX, B_z)*
    nonce$_A$ = random.randint(5, maximum_int);
    tried_ONE_already = False;
    while $\neg h(nonce_{B_z} \wedge TX) \Rightarrow$ *valid* B_z **do**
        nonce$_{B_z}$ = Collatz_Update(*nonce$_{B_z}$*, tried_ONE_already);
        if *nonce$_{B_z}$ == 1* **then**
            tried_ONE_already == True
        end
    end
return B_z
Start:
Construct forged TX_2;
Modify on TX_1 in B_z ($\Rightarrow TX'_1$) to make TX_2 valid;
B'_z = Forge_block (TX'_1, B_z);
B_{z+1} = Forge_block (TX_2, B_{z+1})

sented in Figure 2 and detailed in Figure 3. In the table, we further added the average success rates for $r \geq 8$ (since this was the least difficulty used since the launch of Bitcoin). We then compared the computed average with its relevant q value to check if the proposed approach provided higher attack success rate than theoretical expectations.

It is easily notable from the results that Algorithm 1 provides less success attack rate, than the probability computed by the original Bitcoin calculations. This confirms that Algorithm 1 is not a suitable candidate to attack a Bitcoin-based network in its current form. Our results also confirm that Equation 9 holds regardless of the value of Ω even when the first assumption is violated (i.e. in our case: partially). On the other hand, increasing the difficulty does indeed lower the attack success probability, using Algorithm 1 in its current form, equally to the decrement in honest mining probability as we obtained a Poisson Distribution.

If Algorithm 1 was optimized by answering RQ4, a Random Brute-force Attack on Bitcoin can be successful. As a result, honest nodes would maintain lesser probability to find a puzzle solution through time (due to the difficulty increment), while the attacker's success probability Q needs to be redefined. The work of Bhattacharjee et al. [43] would be a good start to find an answer for RQ4.

It is also noticeable that the experimental attack success rates conforms with the probability distribution of the Birthday Paradox calculations (Equation 15) more than they do with the typically expected linear distribution (Equation 14 deployed in Equation 6). See Figure 1 for demonstration.

5.2. Partial Preimage attack

This attack is a special-case, and relaxed version of the Preimage attack. Given r bits of a hash value $h(m1)$, find an input $m2$ such that $h(m2)$ matches $h(m1)$ in the specified r bits [44]. This attack is claimed in the literature to be inefficient for SHA-256, due to the high output distribution randomness of SHA-256. Mapping this attack into Bitcoin mining, Equation 11 defines r, while $h(m1)$ can be any hash that consists of r leading Zeros (i.e. any element in the set H'). One approach to run this attack is to find $h(m2) = (0 \times r) + s$ of a constant $m2$, by only searching for a correct $s \in H$ instead of searching in all 2^{256} possible outputs of SHA-256.

Few previous works investigated such interesting approach. For example, J. Heusser [37, 45] proposed, in the year 2013, encoding the nonce search as a decision problem solved by a SAT solver [46] in such a way that a satisfiable $m2$ contains a valid nonce. The key ingredients in the algorithm are a non-deterministic nonce, and the ability to take advantage of the known structure of a valid $h(m2)$ referring to an element in H' using assume statements. Although it was not formalized, the method's efficiency was reported to be negatively proportional to Ω (compared to classical CPUs [45]) and, hence, it (potentially) gets more efficient with increasing Bitcoin's difficulty. A miner that uses this approach shall then find puzzle solutions quicker than a miner that uses the classical Brute-force mining approach. Note that this approach implies that the attacker would use a different mining mechanism and (partially) different resources. As a result, a successful attack using this approach violates both the first and the second assumptions.

Other works investigated the utilization of SAT solvers for other purposes, such as in [47] where cryptanalysis were conducted against MD4 and MD5 hash functions, and for other applications [48, 49]. However, there is an unjustifiable lack of research work in, specifically, exploiting SAT solvers to perform partial preimage attacks on PoW-based solutions. In this subsection, we investigate the potential of the approach presented in [37, 45] in particular. If this approach was provably implemented, Q can be redefined as in Equation 17.

$$Q' = \frac{q\varpi\, \xi_{HC}}{\xi_{H'}} \tag{17}$$

Remark 4. *We have optimistically extracted Equation 17 from Equation 14, which provides less probability for a successful attack. If Equation 17 is to be extracted from Equation 16, it would provide higher probability for the attack.*

The default setting, of an honest-majority based Bitcoin network, increases the difficulty as generating new blocks gets

Difficulty (r)	Attacker Succes Rate per q value								
q =	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
1	0.49	0.52	0.35	0.43	0.59	0.51	0.47	0.42	0.56
2	0.26	0.3	0.29	0.27	0.34	0.29	0.3	0.32	0.43
3	0.11	0.18	0.09	0.19	0.18	0.26	0.21	0.24	0.36
4	0.04	0.07	0.06	0.14	0.19	0.18	0.24	0.35	0.31
5	0.03	0.06	0.09	0.19	0.11	0.15	0.16	0.24	0.26
6	0.02	0.07	0.12	0.07	0.15	0.17	0.19	0.31	0.32
7	0	0.09	0.13	0.11	0.22	0.18	0.24	0.27	0.3
8	0.03	0.08	0.12	0.13	0.22	0.28	0.28	0.34	0.31
9	0.01	0.07	0.14	0.11	0.2	0.24	0.24	0.4	0.35
10	0.04	0.07	0.12	0.18	0.21	0.19	0.24	0.28	0.37
11	0.03	0.14	0.14	0.24	0.24	0.26	0.33	0.36	0.36
12	0.02	0.06	0.05	0.19	0.22	0.23	0.19	0.24	0.35
13	0.05	0.04	0.15	0.13	0.23	0.19	0.24	0.32	0.24
14	0.05	0.07	0.07	0.13	0.19	0.23	0.27	0.27	0.28
average for r >= 8	**0.033**	**0.0757**	**0.113**	**0.1586**	**0.216**	**0.231**	**0.256**	**0.316**	**0.3229**
average <= q ?	*TRUE*	*TRUE*	*TRUE*	*TRUE*	*TRUE*	*TRUE*	*TRUE*	*TRUE*	*TRUE*

Figure 3: Attack success rates when using Algorithm 1 with different parameterization of puzzle difficulty and portion controlled by the attacker (q). The color scale (Green-Yellow-Red) is correlated with the attack success rate value on a hypothetical fuzzy spectrum (Low-Medium-High).

faster. If the probability of a successful attack, using the approach discussed above, gets higher with increased difficulty, the increment in difficulty might then lead to swift collapse of the Bitcoin's system security before a hard fork of the core protocol is available. The hard fork shall be able to detect such adversary behaviour within the network and dynamically adjust the difficulty modification mechanism. However, even with such adjustment, the majority of honest nodes converting to this method might lead to decreasing the difficulty, which makes the classical mining mechanism arise as an attack in a scenario similar to an Oscillating Universe [50]. A proven secure, practical and comprehensive method, for detecting and addressing such attack is, then, urgently needed.

We can predict the implications of using such approach for attacking the Bitcoin system by comparing the honest majority success probability (Equation 7) versus the attacker's success probability (Equation 17). Consequently, an attacker can have an advantage over honest nodes as follows:

$$\begin{aligned} E \;&<?\; Q' \\ &\Rightarrow p \times \frac{\xi_{HC}}{\xi_{H'} + \xi_{H''}} \;<?\; q \times \frac{\varpi \xi_{HC}}{\xi_{H'}} \\ &\Rightarrow \frac{1}{2^{256}} \;<?\; \frac{q}{p} \times \frac{\varpi}{2^{4\Omega}} \end{aligned}$$

which gives:

$$\frac{1}{\varpi \times 2^{4r}} < \frac{q}{p} \tag{18}$$

Inequality 18 is, in fact, a direct utilization example of Equation 10. This inequality represents a condition upon which a successful attack (using the approach discussed above) can be conducted. Only for attack probability demonstration purposes[4] we take the pool that controls the highest hash rate per year as K, and the remaining pooled and non-pooled miners as M. Assuming that $\varpi = 1$ (i.e. $\xi_H \leq \xi_C$), we present in Figure 4 a historical data[5] for p, q and the accompanying values of q/p. More in-depth analysis regarding mining pools are provided in [51]. It can be noticed from the figure that inequality 18 holds indeed since the year 2010 (thus associated with the emergence of mining pools), while the maximum portion, historically ever, controlled by a single mining pool, was reached in the year 2013.

Nevertheless, even when inequality 18 holds, the attacker may not be able to catch up with the honest majority with a significantly dominating computing capacity controlled by an honest majority. That is, such dominance shall compensate for the system's decreasing ability of attack tolerance due to

[4] We are not accusing any party as our intention is to clarify our math analysis for the presented security threat.

[5] BTC GUILD: btc.com/stats/pool?percent_mode=2013#pool-history

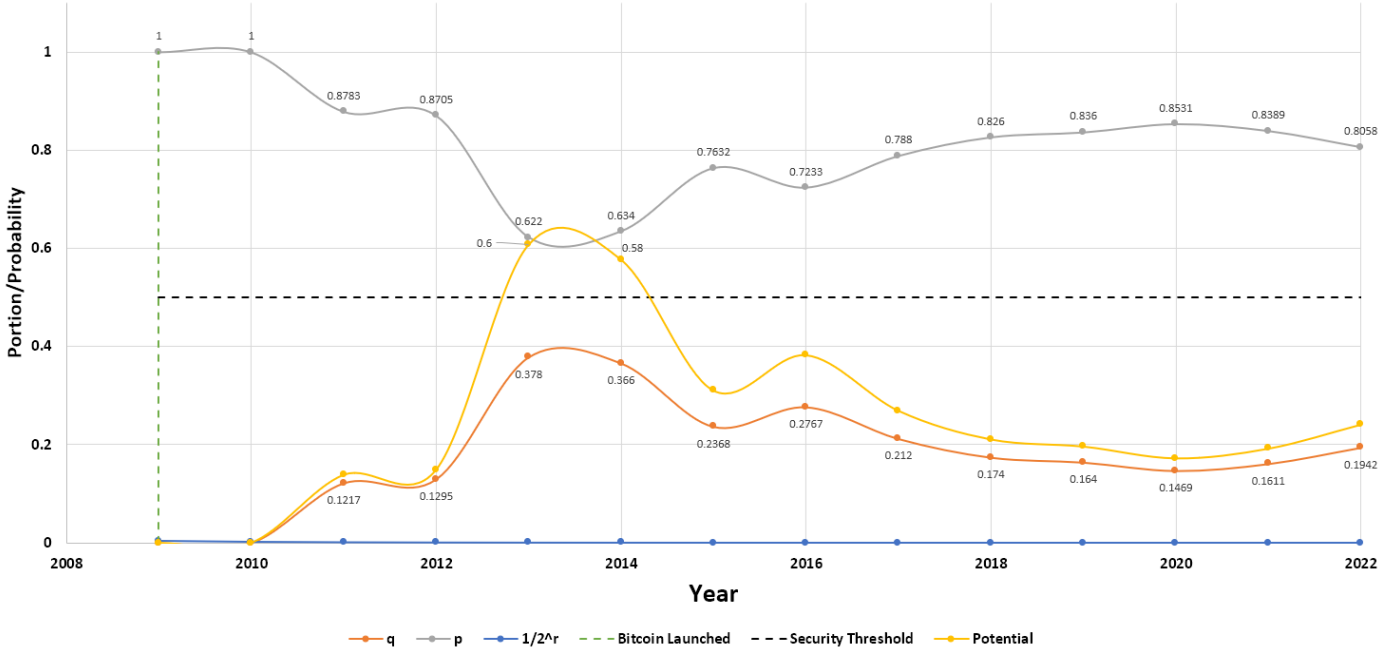


Figure 4: Historical data for maximum portion controlled by the dominating mining pool per year

the increment of the difficulty. It would be beneficial, then, to benchmark and redefine a threshold for such a case. The aim of this benchmark is to provide exact critical r and q values upon which an attacker can successfully run the attack. Since violating the third assumption would not require further violations to control the network, we will only compute Q' values while the third assumption still holds (i.e. using Equation 17 while $p > q$). As long as $Q' < p$, the discussed partial pre-image attack should not be considered as a major threat. Note that we assume that Φ for the honest majority equals 1, which is the best case possible for M.

We test two cases where $\xi_{H'}$ equals 2^{Ω} (hypothetical) and $2^{4\Omega}$ (realistic). Results for computing Q' in those cases are presented in Figures 5 and 6, where the red arrows point to the correlated p value per each q curve.

To check where the Bitcoin system currently stands against the presented attack, the puzzle difficulty r is currently 19 digits (out of 64 hash digits as of May-2022[6]) and is been around this level for almost 3 years now [52]. To successfully run the discussed attack, the attacker would need to further violate the third assumption, which has never been the case, or wait until r exceeds a critical value. It is worth noting as well that the analysis of space-time trade-offs [53] for both mining approaches (i.e. Brute-force and partial pre-image) are of high importance. Specifically, ϖ and q in Equation 17 represent the storage capacity and the computational capacity of the attacker, respectively. Although Bitcoin's original proposal assumed unlimited resources (in its Section 11), we expect Q' to be more constrained due to (realistic) limited resources assumption, resulting in further reduction of successful attack probability.

[6]explorer.btc.com/btc/block/737856

In light of the results presented so far, we propose the concepts of ***Critical Difficulty*** ($\hat{r}$), and ***Critical Difficulty per Portion*** $\hat{r}$ (q).

Definition 1. ***Critical Difficulty*** *($\hat{r}$) is the highest PoW difficulty r after which Equation 9 would not hold anymore.*

We state according to our analysis that $\hat{r} = 56$. Referring to Remark 4, $\hat{r}$ value might be different if Equation 17 was extracted from Equation 16. We leave, as an open issue, the determination and proof of the $\hat{r}$ Equation.

As can be noted in Figures 5 and 6, once the puzzle difficulty r exceeds $\hat{r}$, Equation 9 does not hold anymore. Specifically, the following two experimental observations provided in [37] are conditionally proven by our formalization and analysis:

1. The proposed algorithm gets more efficient with increasing Bitcoin difficulty.
2. The search time of the proposed algorithm is not linearly correlated with the nonce range.

Results shown in Figures 5 and 6 prove that, strictly as long as $r > \hat{r}$, the first observation is indeed true. Taking it from another perspective, let the efficiency of the algorithm be defined with reference to the maximum number of nonce values the algorithm needs to try to obtain a solution. Increasing the difficulty r implies the decrement of Ω leading to decreased number of required nonce values to be tested (Equation 12) while the Algorithm's potential to find the solution is maintained (Equation 13).

For the second observation, this is indeed expected since increasing ξ_C simultaneously increases both the search time and the probability to find a solution (see Equations 13 and 17). That is, higher ξ_C requires the deployed solver to search more

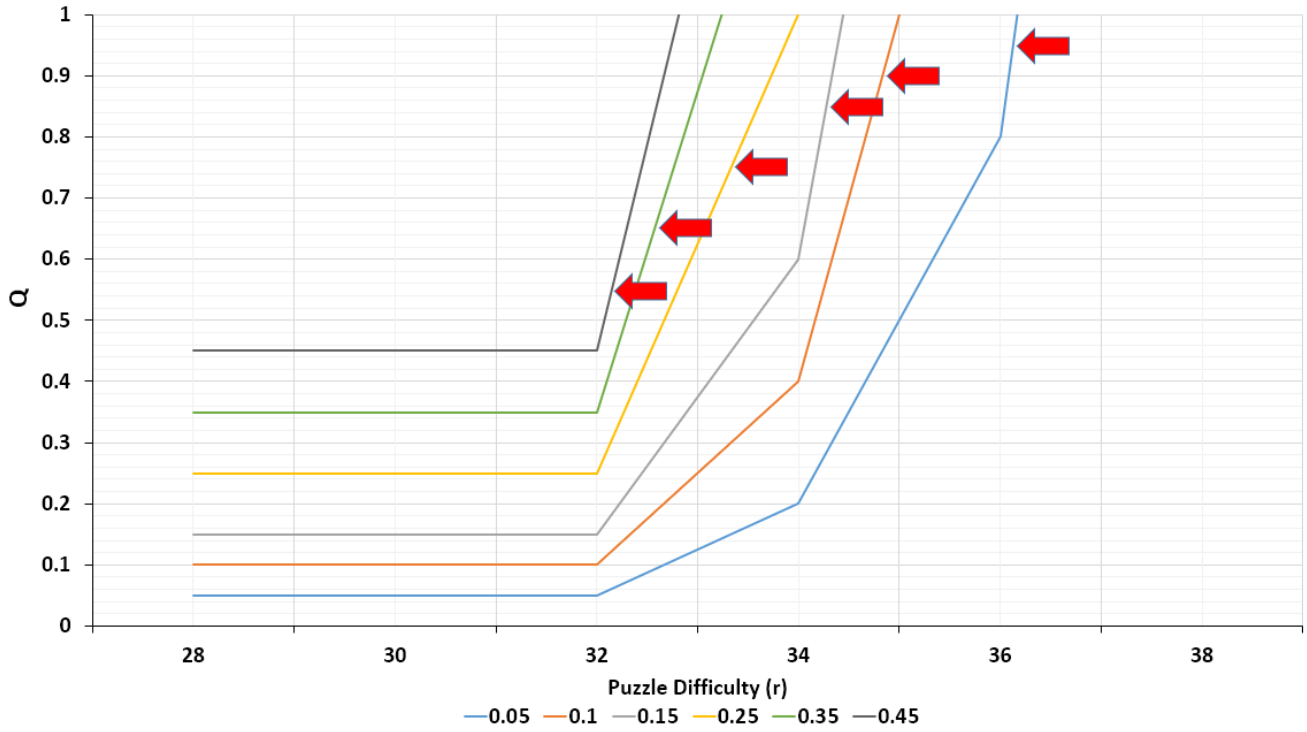


Figure 5: Partial Pre-image Attack probability, on Bitcoin's Proof-of-Work -based system, where $0 < r < 64$, $q \leq 0.45$, $\xi_{H'} = 2^{\Omega}$, and $\varpi = 1$. Red arrows indicate the horizontal grid line for the relevant honest proportion p.

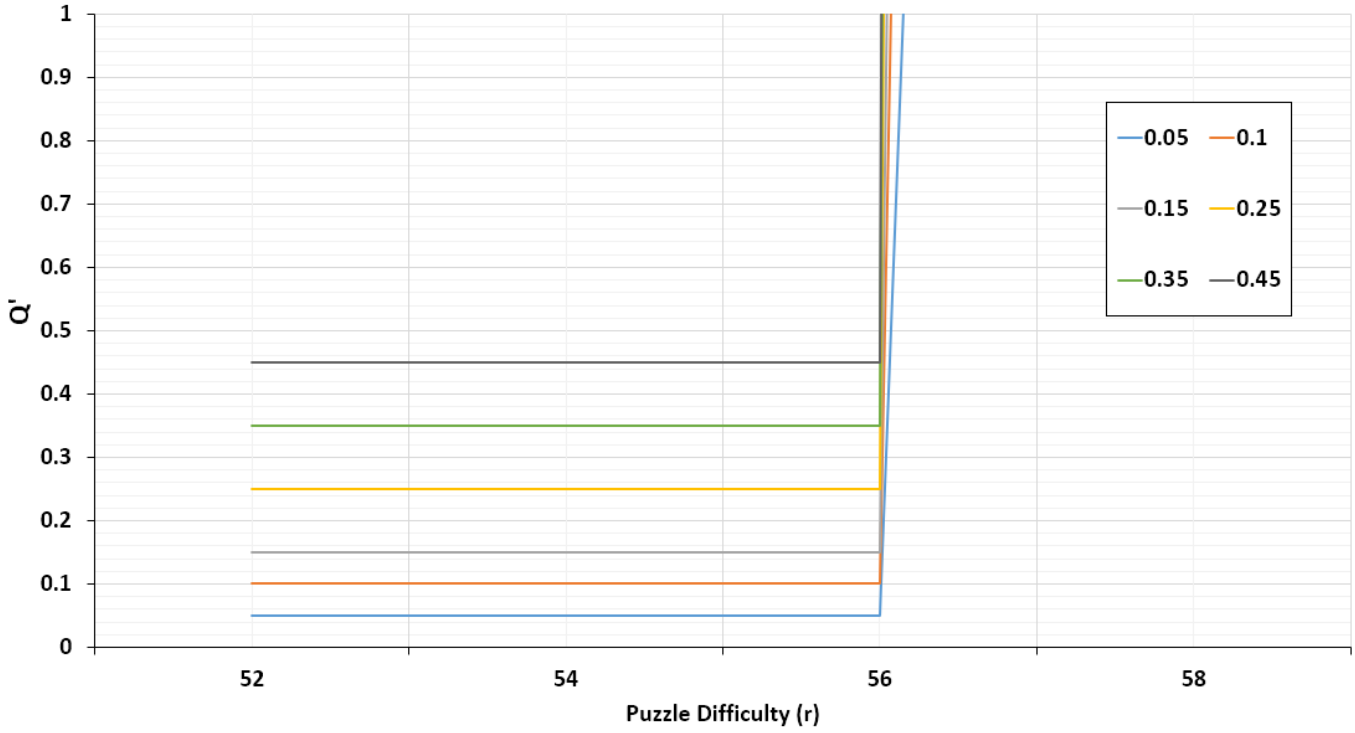


Figure 6: Partial Pre-image Attack probability, on Bitcoin's Proof-of-Work -based system, where $0 < r < 64$, $q \leq 0.45$, $\xi_{H'} = 2^{4\Omega}$, and $\varpi = 1$.

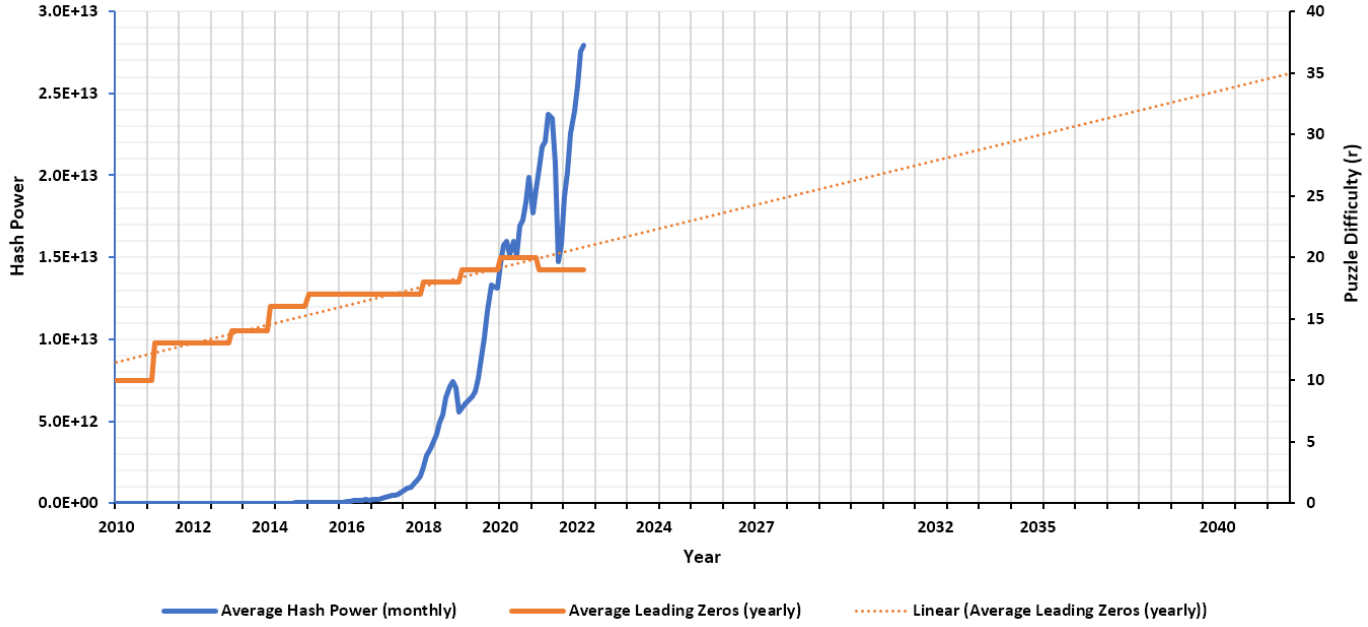


Figure 7: Bitcoin's monthly Hash rate averages (Blue curve correlated with the primary y-axis on the left), yearly Puzzle Difficulty (r) averages (Orange curve correlated with the secondary y-axis on the right) for the period Jan, 2010 – Mar, 2022, and Linear Forecast of r until $r = 35$.

paths for a solution as it uses the DPLL backtracking algorithm which shall consume more processing time. To clarify the correlation, lets assume two hypothetical $C1$ and $C2$ sets with lengths ξ_{C1} and ξ_{C2}, respectively, where $\xi_{C1} < \xi_{C2}$. The search using $C1$ shall take time $T1 < T2$, yet if a MATCH was not found in $C1$ it is more likely to be found sooner using $C2$ as per Equation 13.

To demonstrate how our equations and definitions can be used, we forecast r in order to predict when Bitcoin would reach $\hat{r}$ (for a hypothetical $\hat{r} = 32$) depending on historical evolution of r through time. We extracted the historical data (number of leading zeros (r) and the hash rate of the network, measured from January 2010 until March 2022) using Google's BigQuery[7] database. Using a linear trending approach, we could expect r to reach $\hat{r}$ by the year 2040. Similarly, we can expect that r would reach $\hat{r} = 56$ by the year 2073. Figure 7 presents the data we extracted along with our forecast.

Definition 2. ***Critical Difficulty per Portion*** *($\hat{r}(q)$) is the lowest PoW difficulty r at which Q becomes greater than, or equal to, p.*

Remark 5. *Our definition of* ***Critical Difficulty per Portion*** *$\hat{r}(q)$ assumes that Φ for the honest majority equals 1, which is the best case possible for M.*

Taking the optimistic worst case for K by using Equation 17 (see Remark 4) to calculate Q, the equation of $\hat{r}(q)$ for Bitcoin can be derived as follows:

$$
\begin{aligned}
Q' = p \Rightarrow q \times \frac{2^{\log_2 \xi_C}}{2^{\log_2 \xi_{H'}}} &= p \\
\Rightarrow 2^{\log_2 \xi_C - \log_2 \xi_{H'}} &= p/q \\
\Rightarrow \ln 2^{\log_2 \xi_C - \log_2 \xi_{H'}} &= \ln p/q \\
\Rightarrow (\log_2 \xi_C - \log_2 \xi_{H'}) \times \ln 2 &= \ln p/q \\
\Rightarrow \log_2 \xi_C - \log_2 \xi_{H'} &= \frac{\ln p/q}{\ln 2} \\
\Rightarrow \log_2 \xi_{H'} &= \log_2 \xi_C - \log_2 p/q \\
\Rightarrow \log_2 \xi_{H'} &= \log_2 \frac{q \times \xi_C}{p}
\end{aligned}
$$

Since $\log_2 \xi_{H'} = 256 - 4r$:

$$\hat{r}(q) = (256 - \log_2 \frac{q \times \xi_C}{p})/4 \tag{19}$$

Similar equations can be further derived from Equation 19, e.g. Critical Portion per Difficulty $\hat{q}(r)$ and Critical Difficulty per Portion per Depth of attacked block $\hat{r}(q, z)$. We attempt to use the Ceiling operation in Equation 19 to estimate the first r value at which the inequality $p \leq Q$ is guaranteed valid. However, the Floor operation can be used instead to estimate the highest r value tolerated by the system for a given q. We also used $\log_2 \xi_C = 32$ similarly to its value for honest mining. Note that decreasing $\log_2 \xi_C$ trivially increases $\hat{r}$, which further secures the system and, thus, conforms with the results presented in [37]. Specifically, the experiments presented in [37] configured ξ_C to equal 1000 and 10,000 which gives $\log_2 \xi_C \approx 10$ and 13, respectively. We calculated $\hat{r}(q)$ for $q \in [0.01, 0.5]$ and $\log_2 \xi_C = 32$. The exact and ceiled results are depicted in Figure 8.

[7] https://cloud.google.com/bigquery/public-data/

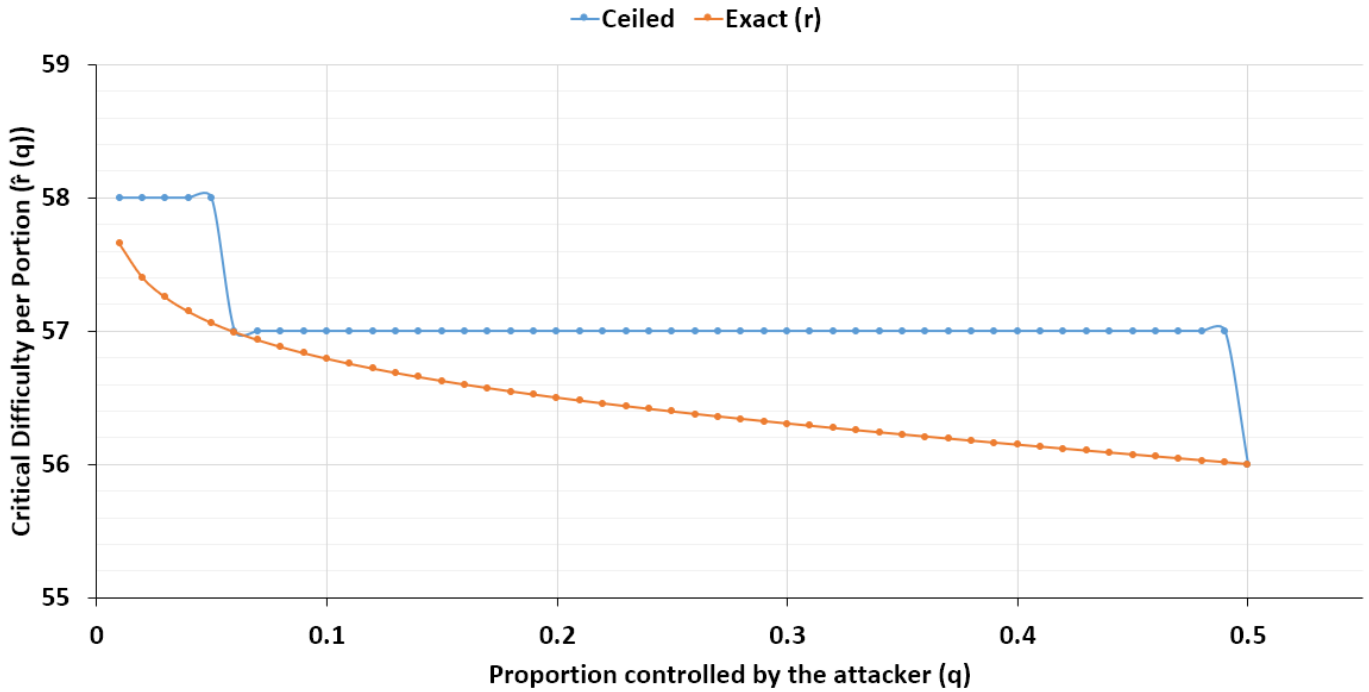


Figure 8: Exact and Ceiled Critical Difficulty per Portion controlled by an attacker $\hat{r}(q)$, where $\log_2 \xi_C = 32$.

In response to RQ 1, it is clear from Equations 17, 18, and 19, that the presented attack is a practical case where Equation 5 cannot be substituted by Equation 9. Additionally, in response to RQ 3, it is clear from the results presented thus far that violating only the first two main assumptions of Bitcoin indeed weakens the security of Bitcoin, and similar, systems, more than originally expected. The presented attack, in particular, is a practical case where violating only those two assumptions, a case that is widely thought to be non-existent or non-practical, can lead to successful security attack on Bitcoin.

5.3. Quantum Attacks

In addition to SHA-256 for securing the ledger against alteration attacks, Bitcoin uses the ECDSA asymmetric cryptography to secure users' data and TXs [54]. Both security measures were shown to be vulnerable against Quantum Computing (QC), which offers fundamentally different solutions to computational problems and enables more efficient problem-solving than what is possible with classical computations [12]. For example, a quantum computer with a clock speed of 100 MHz could factor a 129-bit RSA cipher in a few seconds [55]. QC utilizes the fundamentals of Quantum physics and its definitions and applications to provide more efficient and fast computational paradigms. To directly hit our point, fine-tuned and well equipped QC machines (or Quantum Computers) can be used to decipher secrets, originally encrypted using an asymmetric method, or even to find hash collisions much faster than classical computing methods. Detailed explanation of QC foundations, algorithms, gates, qubits, and countermeasures can be found in [56] and [57].

As argued in [56], it is only a matter of "when" and not "if" QC will become functional as a threat to Bitcoin, and Blockchain technology in general. Authors of [58] further affirmed that a crucial argument in the analysis of Bitcoin security breaks down when QC-based mining is performed. For example, they stated that the chances of a successful QC-based attack, i.e. finding a block using QC, grow quadratically with the number of Grover iterations [59] applied. During the past decade, big IT companies have been spending much time and funds to provide both reliable QC services and QC-resistant cryptographic methods [60]. Examples include IBM [61], Intel [62], Microsoft [63], and Google [64].

Several previous works have investigated the potential of QC attacking the, specifically, Bitcoin mining problem. For instance, Authors of [65] and Authors of [66] studied and explained how the Grover's Algorithm [59] can be used to perform Bitcoin mining. They also well explained QC attacks and they discussed the probability and the profitability of a successful attack. The authors have also discussed using Shor's algorithm [67] to find the private keys from a public key using QC. Specifically, there are a total of 64 rounds of hashing in the SHA256 protocol and each round can be done using an optimized circuit with 683 Toffoli quantum gates [68] (i.e. Controlled Controlled NOT (CCN) gates [55]). In [66], specifically, the number of physical qubits needed to perform Bitcoin mining was defined as a function of overhead in physical qubits, incurred due to quantum error correction, and of the mining difficulty and gate error rate. In [54] and [69], on the other hand, a Quantum attacks resistant protocol, called Commit-Delay-Reveal, was discussed, which allows users to securely move their funds from non-quantum-resistant outputs

to those adhering to a quantum-resistant Digital Signature (DS) scheme. In [58], a mechanism to prevent from quantum mining is proposed, while authors of [70] proved a weak variant of the second conjecture presented in [58].

On the contrary of the above discussed works, authors of [71] proposed a revolutionary QC-based cryptocurrency system. The economic aspects of this system are similar to Bitcoin's, yet it deploys QC for mining instead of PCs and ASICs for classical methods. Perhaps such solution is still not applicable as Quantum Computers are not yet commercialized. However, such solution is trivially expected to be the next cryptocurrency module once Quantum computers are affordable and available for public use. Additionally, such solution provides QC-attacks resistance methods by design, a feature that is not currently available in cryptocurrencies. In [72], several quantum payment schemes were introduced, for implementing prudent contracts—a non-trivial subset of the functionality that a network such as Ethereum provides. The proposed schemes utilize the futuristic affordability vision of Quantum computers for novel quantum-resistant cryptocurrencies and smart contracts.

With reference to the previously presented Bitcoin's major assumptions, QC utilization violates only the second assumption. That is, QC algorithms do search in the whole space of probable solutions, but uses different resources to do so, including the physical superposition phenomena and searching the space reversibly. Specifically, we researched in the literature for answers responding to RQ 5.

RQ 5. *When and how can a Quantum Attack be launched on Bitcoin?*

The "When" part of RQ 5 was recently responded to by [73]. The authors investigated the speed and energy efficiency a quantum computer needs to offer an advantage over classical mining while the third Bitcoin assumption is preserved. To do so, an optimal mining procedure was presented, which was used to show the outperformance of an attacker compared to a classical computer in efficiency (cost per block). A condition, under which this attack can be profitable, is mathematically described[8] by the inequality $Q < Crb$, where Q is the cost of a Grover iteration, C is the cost of a classical hash, r is the quantum miner's speed in Grover iterations per second, and b is a factor that attains its maximum if the quantum miner uses the proposed optimal mining procedure. Similar to threshold benchmarks we proposed earlier for the partial pre-image attacks, the condition provided in [73] presents a threshold benchmark for which quantum mining, and the known security risks associated with it, may arise as a major security threat.

The "How" part of RQ 5 was responded to by several previous works, including specific requirements for each approach. We found that two main types of QC-based attacks can be launched against Bitcoin-based, and similar, systems. The first type uses the Grover's Algorithm and it mainly aims at attacking the hash function security by finding collisions. Any type of attacks on SHA-256 (see Subsection 2.2) can be launched using this algorithm as follows:

- The number of attacker's Qubits and/or Quantum Computers must be sufficient to rival the probability of finding the next block using classical Brute-force [65]. In other words, the third assumption needs to be violated as well.
- A Quantum-based method for Bitcoin mining can be executed in time $O(\sqrt{2^{\Omega}})$, which is much less than classical Brute-force settings [58]. However, it was stated that the size of the quantum computer needs to be $\approx 10^4$ qubits in order to provide such attack efficiency.

The second type uses Shor's Algorithm for reusing addresses, double spending attacks, or live TX hijacking. Aside from the recommendation of most works to use different addresses and keys for each TX, several quantum safe alternatives, for the ECDSA signature scheme used in Bitcoin, were proposed [74]. The most promising proposed approaches include the Winternitz One-Time Signature Scheme (W-OTS+) [75] and the SPHINCS+ [76] method. Those, two methods, specifically, were detailed in [77] and compared to ECDSA and RSA schemes in [78]. For the second type of QC attacks (i.e. using Shor's algorithm), we found the following measures to be the most relevant responses to RQ 5 (each with its source reference):

- 1500 qubits are required and 6×10^9 one-qubit additions are needed (Each one-qubit addition takes 9 quantum gates) in order to successfully attack Bitcoin [79].
- 2330 qubits are needed and 1.26×10^{11} Toffoli gate operations are required in order to successfully attack Bitcoin [80].
- for 10GHz clock speed and error rate of 10^{-5} , Bitcoin signatures can be cracked in 30 minutes using 485550 qubits [66], according to which, signatures are expected to be easily broken, in time less than the Bitcoin block time, by the year 2027.
- The number of qubits required for the Bitcoin's case is roughly 1536 qubits [79] [81].

As can be noticed in both types of QC-based attacks, thousands of qubits are required to launch a profitable Bitcoin mining or hijacking its TXs. As of May 2022, the leading giant in QC; IBM, announced its biggest ever Quantum computer that consists of 433 qubits [82], which still does not violate the third assumption of Bitcoin. To summarize, although QC-based attacks violate the second assumption, quantum computers cannot successfully attack Bitcoin unless they also violate the third assumption.

5.4. Bitcoin Sharding

Sharding is a type of database partitioning technique that separates a very large database into much smaller, faster, more easily managed parts called data shards [83]. Technically, sharding

[8] We kindly ask the Reader not to confuse these notations with our current work's notations, as we used similar ones presented in reference [73].

is a synonym for horizontal partitioning, which makes a large database more manageable and efficient. Following the notations detailed in Subsection 2.1, the key idea of BC sharding is to partition V (which are formed initially as one giant shard) into a set $\Gamma = \{Shard_1, Shard_2, ..Shard_\ell\}$. Each $Shard_i \in \Gamma$ consists of $N_{Shard_i} < N$ nodes until condition 20 is satisfied. Each shard processes a disjoint set of TXs, yet all shards utilize the same CA, leading to increased overall system throughput. As described in [84], ℓ grows linearly with both T and N. Denoting the number of adversary nodes in $Shard_i$ as $f_{Shard_i} <= f$, a sharding protocol outputs a set Γ leading to State 21.

$$\sum_{i=1}^{\ell} N_{Shard_i} = N \tag{20}$$

$$\sum_{i=1}^{\ell} f_{Shard_i} = f \tag{21}$$

Typically, a solution for a problem in a distributed system, such as sharding Bitcoin, should consider three main, strongly-connected factors affecting the optimization of that solution. Those three factors are decentralization, scalability, and security. Although privacy issues/discussions are technically different than security issues/discussions, privacy preservation is usually considered as a subdomain within the security discussion. Accordingly, a suitable sharding protocol for Bitcoin can be characterized by answering the following Research Questions:

RQ 6. *(**Decentralization**) Which system element should be responsible for computing the set* Γ*?*

RQ 7. *(**Scalability**) What is the suitable approach to compute the set* Γ*?*

RQ 8. *(**Security and Privacy**) Must Bitcoin's* Ψ *be maintained in each shard? If Yes, then how?*

5.4.1. Decentralization

The decentralization level, provided in the permissionless public model of Bitcoin, is foundational for gaining trust and reliability. The more entities participating in the protocol, the more trusted the system becomes. To clarify, a sharded version of any network is obviously less decentralized than its non-sharded network. Since ℓ (number of shards) is by definition less than N (number of nodes), the number of entities maintaining the global view of the sharded network's ledger is much less than those which maintain the ledger in the network when it is not sharded.

On the other hand, since each shard processes different TXs than those processed by other shards, the number of validations and confirmations granted per TX is much less than confirmations granted per TX in the non-sharded version of the network. In other words, trust in the finality of a given TX in a sharded network is less than the trust in the finality of the same TX in the same network that is non-sharded.

Furthermore, since a permissionless public BC implies that any node participating in the network might behave dishonestly, a foundational question for sharding Bitcoin is RQ 6. We have few answers to choose from:

1. A Trusted Third Party (TTP): which implies the inclusion of a special node(s) that has advanced access permissions to private data of the miners, including their computational capacity, IP-addresses, neighbors, bandwidth, etc. This imposes a critical privacy concern regarding the capability of the TTP to build and misuse global information about the network. Furthermore, a dishonest TTP is assumed possible in Bitcoin, thus was (and still) not used.
2. An Elected Leader: Leader election problem in fully distributed systems was studied in several previous works. Although it is hard to address, once addressed in public permissionless BCs either privacy violations, or very high complexities are usually expected [85]. Even if agreed-on to be used, incentivization mechanisms need to be implemented, and high complexity issues must be solved.
3. Dynamic Selection of Leader Shards: this is a hybrid version of the the above two methods. Instead of electing a single leader to compute set Γ, a whole shard is elected which utilizes a CA to agree on a sharded version. Although this option solves the problems faced by a TTP utilization, it adds one extra layer of complexity to the single leader election problem (i.e. complexity associated with the CA). Furthermore, it is obvious that in such settings, a small group in V controls the logical communications amongst all V elements. For this, the majority of nodes within a shard must be honest or the system security would crash (discussed later).

It is noticeable by now how sharding indeed deviates the Bitcoin network towards centralization. Nevertheless, with adequate tuning of the sharding parameterization, this should not be a big concern knowing that there are tens of thousands of nodes. However, high scalability gained from sharding the network must be convincing for lowering the decentralization level. Assuming that there is a sharding protocol that preserves equivalent level of security with high level of scalability, the leader shard option seems to be the most appropriate.

5.4.2. Scalability

A sharding protocol is theoretically expected to enhance a BC-based solution in terms of scalability. That is, the scalability of the BC-based solution is usually benchmarked by the overall throughput of the system while increasing N. Since processing tasks is distributed among shards, new TXs and blocks can be processed in parallel. Accordingly, increasing the number of shards (ℓ) should increase the overall throughput of the system regardless of the approach of nodes distribution among shards. Nevertheless, ℓ configuration must not be too high so that each shard would consist of one node, nor too low so that the scalability is not much enhanced. Thus, the first configuration needed is an adequate number of shards per a given V.

Since the aim of sharding is to process TXs and blocks faster, we do also care about shards' individual throughput level. The more shards with high throughput, the higher the throughput of the whole system. To achieve an optimal throughput in a

given shard, the weight or the diameter of the network within needs to be minimized. As in non-sharded networks, the lower the weight or the diameter, the sooner data are propagated and, thus, the faster the processing of data. This, in fact, is a well known optimization problem in Graph Theory, namely Graph Partitioning Problem (GPP) [86], whose objective function is to minimize the average weight within shards.

As all optimization problems, GPP can be formalized using heuristics [87], meta-heuristics [88], or linear programming [86]. Aside from meta-heuristics, other formalization approaches impose high computational complexity to solve GPP. However, meta-heuristics can not provide optimum solutions, nor can be controlled to a certain level of optimization. With considering our previous recommendation regarding the system element to carry on computing the set Γ, we do recommend a meta-heuristic approach to solve the problem since it is fast and requires low computational capacity. However, the optimization criteria is not necessarily accurate. Detailed analysis and comparison between optimization criteria may reveal that minimizing the average weights of shards does not provide optimal scalability. For example, investigating the minimization of average computing capacity of shards might be a good research direction. The multi-objective minimization of average computing capacities together with average weights could be even a better approach (e.g. using a multi-objective ACO [89] or MOHEFT [90]).

As will be discussed later, state-of-the-art solutions do not optimize node distribution as they randomly distribute them among shards. Accordingly, those solutions only partially benefit from sharding the network as TXs are processed in parallel logical threads (i.e. shards) without the optimization of those threads.

5.4.3. Security and Privacy

The security-related *agreement* property in a given sharded BC is benchmarked with reference to the previously discussed Ψ benchmark depending on the CA used. In Bitcoin, Ψ is determined with reference to Inequality 4. If the Bitcoin network is to be sharded, each shard must provide a level of agreement that is equivalent to the agreement level of Bitcoin when it is not sharded. Specifically, Condition 22 must always hold.

$$q_{Shard_i} < \frac{T_{Shard_i}}{2} \quad \forall \quad Shard_i \in \Gamma \tag{22}$$

The requirement to maintain Ψ within each shard is necessary so that the outputs of the shard, i.e. confirmed TXs, agreed-on chain version, and the set Γ in case of our recommendation, are reliable for the system. Since each shard processes a different set of TXs, an inter-shard protocol merges all distinct chain versions produced by each shard into one global DL. Merging approaches have been discussed in the literature and are out of the scope of our work. However, Bitcoin's sharding protocol should be able to maintain Condition 22 simply because a fraud sub-chain would not be detected as a fraud once it is confirmed by a shard's majority. If Ψ is not guaranteed on the shard level then no TX can be proven valid.

Table 2: Technical details of state-of-the-art sharding solutions.

Sharding Solution	TTP?	Ψ	DS?	Approach
[92]	Yes	$f \leq N/4$	No	Randomized
ELASTICO [93]	Yes	$q \leq 1/4$	Yes	Randomized
Ostraka [94]	Yes	N/A	No	Randomized
SBSCH-GKA [95]	Yes	N/A	Yes	N/A
Cycledger [96]	Yes	$f \leq$ N/3	Yes	Randomized
RepChain [97]	Yes	$f \leq$ N/3	Yes	Randomized
SSHC [98]	Yes	$q \leq 1/3$	Yes	Randomized
RapidChain [99]	No	$f \leq$ N/3	Yes	Cuckoo [100]
Required for Bitcoin	No	$q \leq 1/2$	No	Non-Randomized

In most sharding protocols, randomized sharding is regularly run in order to maintain a high probability that Condition 22 holds. However, such sharding approach waves away the optimal data propagation within, and among, shards. Specifically, this approach results in relatively low throughput per shard, in exchange for (potential) high level of security, resulting in the throughput of some shards to be nearly equivalent to the non-sharded network's. Nonetheless, randomized sharding can never guarantee that condition 22 always holds since adversary nodes are not detected prior, or during, the run of the sharding protocol. Thus, it is not recommended for such approach to be used in public and permissionless BCs [91], such as Bitcoin.

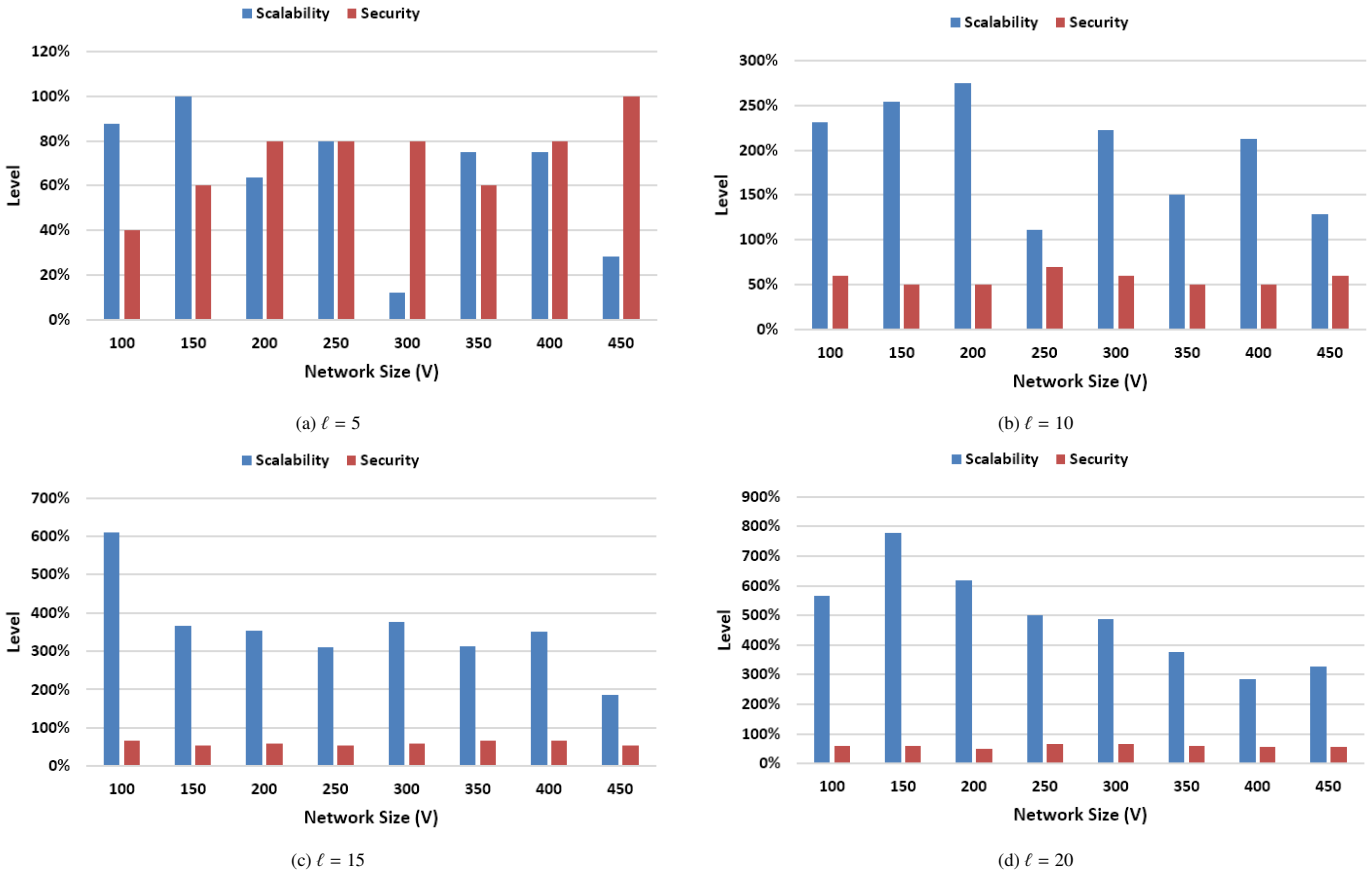
Some previous works suggested using DSs to sign each TX and block, which may indeed solve the security problem. On the other hand, using DSs would further decrease the throughput, due to the relatively high time complexity for signing and verifying, probably resulting in a total throughput even worse than the original throughput of the non-sharded network. Additionally, the utilized DSs represent ID pointers to miners, which raises concerns regarding linkage attacks possibility in public BCs.

5.4.4. Experiments and Discussion

The trade-off between scalability and security in sharded public BCs is an open optimization problem [101]. The utilization of a TTP further deviates the system towards centralization, while the utilization of a leader election raises identity privacy concerns [85]. Decentralization and Privacy were among the few core issues that Bitcoin primarily attempted to solve. As long as this Decentralization-Privacy-Security-Scalability (DePriSS) tetralemma[9] is not solved, sharding the Bitcoin network would always lead to the violation of its third major assumption. That is, even if $p > q$, non of the state-of-the-art BC sharding solutions guarantee Condition 22 to always hold.

In Table 2, we provide brief technical details of recent state-of-the-art works for sharding BCs. The table presents the proposals' requirements for a TTP, or a leader, the initial security assumption, the utilization of DSs, and the GPP sharding approach used. It is obvious from the table that none of the previous works had proposed a suitable sharding solution for Bitcoin, or similar, networks. Even in [94], where the authors claimed that the security level is preserved as is prior to sharding, they had the exception of Bitcoin and left it as a future

[9] or quadrilemma, defined as a difficult choice between four alternatives



(a) $\ell = 5$

(b) $\ell = 10$

(c) $\ell = 15$

(d) $\ell = 20$

Figure 9: Scalability and Security in randomly sharded networks with $q = 0.4$ and different configurations of number of shards (ℓ)

research direction. RapidChain [99] is the closest in its configuration to the needed sharding protocol for Bitcoin. Here, allowing the use of DSs as a relaxation of Privacy and Scalability requirements could not provide a protocol that always guarantees the validity of Equation 22 while $q \leq 1/2$. In short, a Bitcoin sharding protocol should fulfill the requirements presented in Table 2 to address the DePriSS tetralemma.

Let's assume that the Decentralization and Privacy issues were addressed in a sharding protocol that uses a randomization sharding approach. We want to test how such sharding approach contributes to Scalability and Security issues. To do that, we build a simple Bitcoin network simulator using Python 3.9, in which we deployed Networkx[10] and PyMetis[11][12] libraries. To mimic the connection model in Bitcoin, we use a random graph connection model in our experiments, namely Erdos-Renyi model [102], with each node connected to approximately $V/2$ nodes. The weights of edges, representing the transmission time between each two adjacent nodes, was configured randomly as well. The simulator iteratively builds random networks with $V \in [100, 450]$ and $\ell \in [5, 20]$, then it measures the Scalability and Security levels, of the sharded network versions, using Equations 23 and 24. In both of those equations, the higher the result the better. Specifically, the first part of Equation 23 provides the expected throughput outperformance of each individual shard, compared to the non-sharded network (i.e. on average, with reference to the average diameter of shards and of the network). The second part of Equation 23 multiplies this enhancement by the number of shards, resulting in the total expected throughput enhancement of the network due to sharding. Equation 24, on the other hand, tests the security of each shard, using Equation 22, and computes the ratio between the number of secure shards to the total number of shards. A sharding protocol that fulfils the Security requirement should always provide a security level equal to 1.

$$Scalability\_Enhancement = (1 - \frac{Avg\_Shard\_Diameter}{Network\_Diameter}) \times \ell \quad (23)$$

$$Security\_Level = \frac{Number\_Of\_Secure\_Shards}{\ell} \quad (24)$$

Our initial parameterization for all of the non-sharded networks complies with the third assumption (i.e. $p > q$). We ran the described experiments on Google Cloud Platform using a VM of type c2-standard-4 (4 Intel Cascade Lake vCPUs clocked at 3.8GHz with 16GB of RAM), running Ubuntu

[10] networkx.org
[11] metis.readthedocs.io/en/latest
[12] https://github.com/inducer/pymetis

18.04 LTS. Our code is publicly available at Github[13]. The experiments aim to highlight that even if randomly sharding the Bitcoin network would provide higher scalability, it can never guarantee the security unless either DSs were used (thus negatively affects the scalability enhancement), or a TTP (or elected leader) thus further deviates the network towards centralization and less privacy.

The results of our experiments are depicted in Figure 9. As can be noted, increasing the number of shards does indeed significantly increase the scalability even though nodes were distributed randomly among shards. However, Equation 9 was violated in almost all of the cases, which proves that such sharding approach violates Bitcoin's third assumption on the shard level.

6. Conclusion

In this paper, we revisited the Bitcoin's calculations and simplified its main assumptions. Accordingly, we proposed a novel formalization model of the Bitcoin mining problem using the Birthday Paradox. We also presented several attack approaches that could violate Bitcoin's main assumptions. We proposed a Random Brute-force attack algorithm, along with a research question, for which an answer would make the proposed algorithm be the base of a profitable attack. Additionally, we used our formalized model to prove the ability to perform Partial Pre-Image attacks on Bitcoin. We proposed new benchmark thresholds for Bitcoin, including Critical Difficulty and Critical Difficulty per Portion. We applied these benchmarks on historical difficulty data, and found that reaching a puzzle difficulty of 56 leading zeros would allow profitable Partial Pre-Image attacks on Bitcoin. We have also discussed how Quantum Computing can be used to attack Bitcoin, and we provided a comprehensive literature review in this regard. Finally, we discussed the effects of sharding on Bitcoin's security, both theoretically and experimentally. We have showen that sharding the Bitcoin network would always lead to security breaches, unless the network model is deviated towards permissioned instead of permissionless, or the foundational privacy requirements were changed.

Acknowledgment

The research leading to these results has received funding from the National Research, Development and Innovation Office within the framework of the Artificial Intelligence National Laboratory Programme, and from the national project TKP2021-NVA-09 implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260, 2008.

[2] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure information networks*, pages 258–272. Springer, 1999.

[3] FIPS Pub. Secure hash standard (shs). *Fips pub*, 180(4), 2012.

[4] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[5] Henri Massias, X Serret Avila, and J-J Quisquater. Design of a secure timestamping service with minimal trust requirement. In *the 20th Symposium on Information Theory in the Benelux*. Citeseer, 1999.

[6] Nir Kshetri and Jeffrey Voas. Blockchain-enabled e-voting. *Ieee Software*, 35(4):95–99, 2018.

[7] Anton Hasselgren, Katina Kralevska, Danilo Gligoroski, Sindre A Pedersen, and Arild Faxvaag. Blockchain in healthcare and health sciences—a scoping review. *International Journal of Medical Informatics*, 134:104040, 2020.

[8] Mayra Samaniego, Uurtsaikh Jamsrandorj, and Ralph Deters. Blockchain as a service for iot. In *2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, pages 433–436. IEEE, 2016.

[9] Shuai Wang, Liwei Ouyang, Yong Yuan, Xiaochun Ni, Xuan Han, and Fei-Yue Wang. Blockchain-enabled smart contracts: architecture, applications, and future trends. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):2266–2277, 2019.

[10] Tigang Jiang, Hua Fang, and Honggang Wang. Blockchain-based internet of vehicles: Distributed network architecture and performance analysis. *IEEE Internet of Things Journal*, 6(3):4640–4649, 2018.

[11] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In *International Conference on Information Security and Cryptology*, pages 29–40. Springer, 2006.

[12] Laszlo Gyongyosi and Sandor Imre. A survey on quantum computing technology. *Computer Science Review*, 31:51–71, 2019.

[13] Guangsheng Yu, Xu Wang, Kan Yu, Wei Ni, J Andrew Zhang, and Ren Ping Liu. Survey: Sharding in blockchains. *IEEE Access*, 8:14155–14181, 2020.

[14] Melanie Swan. *Blockchain: Blueprint for a new economy*. " O'Reilly Media, Inc.", 2015.

[15] Attila Kertesz and Hamza Baniata. Consistency analysis of distributed ledgers in fog-enhanced blockchains. In *International European Conference on Parallel and Distributed Computing (Euro-Par 2021)*, volume 27, 2021.

[16] Qin Wang, Rujia Li, Shiping Chen, and Yang Xiang. Formal security analysis on dbft protocol of neo. *arXiv preprint arXiv:2105.07459*, 2021.

[17] CISSP Thomas Porter, CCDA CCNP, Michael Gough, et al. *How to cheat at VoIP security*. Syngress, 2011.

[18] Leighton Johnson. *Security controls evaluation, testing, and assessment handbook*. Academic Press, 2019.

[19] Marc Martinus Jacobus Stevens. *Attacks on hash functions and applications*. Leiden University, 2012.

[20] Yu Sasaki, Lei Wang, and Kazumaro Aoki. Preimage attacks on 41-step sha-256 and 46-step sha-512. *Cryptology ePrint Archive*, 2009.

[21] Takanori Isobe and Kyoji Shibutani. Preimage attacks on reduced tiger and sha-2. In *International Workshop on Fast Software Encryption*, pages 139–155. Springer, 2009.

[22] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2 n work. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 474–490. Springer, 2005.

[23] Dan Michael A Cortez, Ariel M Sison, and Ruji P Medina. Cryptographic randomness test of the modified hashing function of sha256 to address length extension attack. In *Proceedings of the 2020 8th International Conference on Communications and Broadband Networking*, pages 24–28, 2020.

[24] L Bošnjak, J Sreš, and Bosnjak Brumen. Brute-force and dictionary attack on hashed real-world passwords. In *2018 41st international convention on information and communication technology, electronics and microelectronics (mipro)*, pages 1161–1166. IEEE, 2018.

[25] Huaqun Guo and Xingjie Yu. A survey on blockchain technology and its

[13]

security. *Blockchain: Research and Applications*, page 100067, 2022.
[26] Henri Gilbert and Helena Handschuh. Security analysis of sha-256 and sisters. In *International workshop on selected areas in cryptography*, pages 175–193. Springer, 2003.
[27] Marcio Juliato and Catherine Gebotys. Seu-resistant sha-256 design for security in satellites. In *2008 10th International Workshop on Signal Processing for Space Communications*, pages 1–7. IEEE, 2008.
[28] H Handschub and Henri Gilbert. Evaluation report security level of cryptography–sha-256. *Journal of Women s Health*, 2002.
[29] Robert Benkoczi, Daya Gaur, Naya Nagy, Marius Nagy, and Shahadat Hossain. Quantum bitcoin mining. *Entropy*, 24(3):323, 2022.
[30] A Pinar Ozisik and Brian Neil Levine. An explanation of nakamoto's analysis of double-spend attacks. *arXiv preprint arXiv:1701.03977*, 2017.
[31] Jinmin Zhong and Xuejia Lai. Preimage attacks on reduced dha-256. *Cryptology ePrint Archive*, 2009.
[32] Himanshu N Bhonge, Monish K Ambat, and BR Chandavarkar. An experimental evaluation of sha-512 for different modes of operation. In *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2020.
[33] Gary Rowe. Has sha256 been broken by tread-well stanton dupont? [online].[accessed 2022-05-13]. *https://crypto.stackexchange.com/questions/74251/has-sha256-been-broken-by-treadwell-stanton-dupont*, 2019.
[34] Joseph J Kearney and Carlos A Perez-Delgado. Vulnerability of blockchain technologies to quantum attacks. *Array*, 10:100065, 2021.
[35] Richard Preston. Applying grover's algorithm to hash functions: A soft-ware perspective. *arXiv preprint arXiv:2202.10982*, 2022.
[36] miningpoolstats.stream. List of mining pools [online].[accessed 2022-05-13]. 2022.
[37] Jonathan Heusser. Sat solving-an alternative to brute force bitcoin mining. Technical report, Technical Report. https://jheusser. github. io/2013/02/03/satcoin. html, 2013.
[38] Hong Guo, Wenzhuo Tang, Yu Liu, and Wei Wei. Truly random number generation based on measurement of phase noise of a laser. *Physical Review E*, 81(5):051137, 2010.
[39] Jeffrey C Lagarias. The 3 x+ 1 problem and its generalizations. *The American Mathematical Monthly*, 92(1):3–23, 1985.
[40] José A Tenreiro Machado, Alexandra Galhano, and Daniel Cao Labora. A clustering perspective of the collatz conjecture. *Mathematics*, 9(4):314, 2021.
[41] David Xu and Dan E Tamir. Pseudo-random number generators based on the collatz conjecture. *International Journal of Information Technology*, 11(3):453–459, 2019.
[42] Alan Kaminsky. Testing the randomness of cryptographic function mappings. *Cryptology ePrint Archive*, 2019.
[43] Kamalika Bhattacharjee and Sukanta Das. A search for good pseudo-random number generators: Survey and empirical studies. *Computer Science Review*, 45:100471, 2022.
[44] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Security evaluation of sha-224, sha-512/224, and sha-512/256. *Institute for Applied Information Processing and Communications, Graz University of Technology*, 2015.
[45] Norbert Manthey and Jonathan Heusser. Satcoin–bitcoin mining via sat. *SAT COMPETITION 2018*, page 67, 2018.
[46] Weiwei Gong and Xu Zhou. A survey of sat solver. In *AIP Conference Proceedings*, volume 1836, page 020059. AIP Publishing LLC, 2017.
[47] Ilya Mironov and Lintao Zhang. Applications of sat solvers to cryptanalysis of hash functions. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 102–115. Springer, 2006.
[48] Fabio Massacci. Using walk-sat and rel-sat for cryptographic key search. In *IJCAI*, volume 99, pages 290–295, 1999.
[49] Benjamin W Bloom. Sat solver attacks on cubehash. Technical report, Citeseer, 2010.
[50] Ruth Durrer and Joachim Laukenmann. The oscillating universe: an alternative to inflation. *Classical and Quantum Gravity*, 13(5):1069, 1996.
[51] Natkamon Tovanich, Nicolas Soulié, Nicolas Heulot, and Petra Isenberg. The evolution of mining pools and miners' behaviors in the bitcoin blockchain. *IEEE Transactions on Network and Service Management*, 2022.
[52] Henri T Heinonen and Alexander Semenov. Recycling hashes from reversible bitcoin mining to seed pseudorandom number generators. In *International Conference on Blockchain*, pages 103–117. Springer, 2021.
[53] Vlad Gheorghiu and Michele Mosca. Benchmarking the quantum cryptanalysis of symmetric, public-key and hash-based cryptographic schemes. *arXiv preprint arXiv:1902.02332*, 2019.
[54] Iain Stewart, Daniel Ilie, Alexei Zamyatin, Sam Werner, MF Torshizi, and William J Knottenbelt. Committing to quantum resistance: a slow defence for bitcoin against a fast quantum computing attack. *Royal Society open science*, 5(6):180410, 2018.
[55] Tony Hey. Quantum computing: an introduction. *Computing & Control Engineering Journal*, 10(3):105–112, 1999.
[56] Dragos Ilie, William Knottenbelt, and Kin Leung. Making bitcoin quantum resistant. 2018.
[57] Alexandru Cojocaru, Juan Garay, Aggelos Kiayias, Fang Song, and Petros Wallden. The bitcoin backbone protocol against quantum adversaries. *Cryptology ePrint Archive*, 2019.
[58] Or Sattath. On the insecurity of quantum bitcoin mining. *International Journal of Information Security*, 19(3):291–302, 2020.
[59] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.
[60] Noah Anhao. Bitcoin post-quantum, 2018.
[61] IBM Q Experience. quantum-computing.ibm.com. Accessed: 2022-06-14.
[62] Intel Quantum Computing Service. intel.com/content/www/us/en /research/quantum-computing.html. Accessed: 2022-06-14.
[63] Microsoft Quantum Computing Service. microsoft.com/en-us/research/research-area/quantum-computing. Accessed: 2022-06-14.
[64] Masoud Mohseni, Peter Read, Hartmut Neven, Sergio Boixo, Vasil Denchev, Ryan Babbush, Austin Fowler, Vadim Smelyanskiy, and John Martinis. Commercialize quantum technologies in five years. *Nature*, 543(7644):171–174, 2017.
[65] Louis Tessler and Tim Byrnes. Bitcoin and quantum computing. *arXiv preprint arXiv:1711.04235*, 2017.
[66] Divesh Aggarwal, Gavin K Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on bitcoin, and how to protect against them. *arXiv preprint arXiv:1710.10377*, 2017.
[67] Peter W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.
[68] Alex Parent, Martin Roetteler, and Krysta M Svore. Reversible circuit compilation with space constraints. *arXiv preprint arXiv:1510.00377*, 2015.
[69] Dragos I Ilie, William J Knottenbelt, and Iain D Stewart. Committing to quantum resistance, better: A speed-and-risk-configurable defence for bitcoin against a fast quantum computing attack. In *Mathematical Research for Blockchain Economy*, pages 117–132. Springer, 2020.
[70] Miklos Santha, Troy Lee, and Maharshi Ray. Strategies for quantum races. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*, 2019.
[71] Jonathan Jogenfors. Quantum bitcoin: an anonymous, distributed, and secure currency secured by the no-cloning theorem of quantum mechanics. In *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 245–252. IEEE, 2019.
[72] Or Sattath. Quantum prudent contracts with applications to bitcoin. *arXiv preprint arXiv:2204.12806*, 2022.
[73] Robert R Nerem and Daya R Gaur. Conditions for advantageous quantum bitcoin mining. *arXiv preprint arXiv:2110.00878*, 2021.
[74] Stephen Holmes and Liqun Chen. Assessment of quantum threat to bitcoin and derived cryptocurrencies. *Cryptology ePrint Archive*, 2021.
[75] Andreas Hülsing. W-ots+–shorter signatures for hash-based signature schemes. In *International Conference on Cryptology in Africa*, pages 173–188. Springer, 2013.
[76] Daniel J Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2129–2146, 2019.
[77] Trung van Trinh. Quantum-safe bitcoin. Master's thesis, 2020.
[78] Moses Dogonyaro Noel, Onomza Victor Waziri, Muhammad Shafii Abdulhamid, Adebayo Joseph Ojeniyi, and Malvis Ugonna Okoro. Com-

parative analysis of classical and post-quantum digital signature algorithms used in bitcoin transactions. In *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, pages 1–6. IEEE, 2020.

[79] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *arXiv preprint quant-ph/0301141*, 2003.

[80] Martin Roetteler, Michael Naehrig, Krysta M Svore, and Kristin Lauter. Quantum resource estimates for computing elliptic curve discrete logarithms. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 241–270. Springer, 2017.

[81] Neeraj Samtani. How would quantum computing impact the security of bitcoin by enhancing our ability to solve the elliptic curve discrete logarithm problem? *Available at SSRN 3232101*, 2018.

[82] IBM Quantum Computer. ibm.com/quantum. Accessed: 2022-06-14.

[83] Yimeng Liu, Yizhi Wang, and Yi Jin. Research on the improvement of mongodb auto-sharding in cloud environment. In *2012 7th international conference on Computer science & education (ICCSE)*, pages 851–854. IEEE, 2012.

[84] Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 41–61, 2019.

[85] Hamza Baniata, Ahmad Anaqreh, and Attila Kertesz. DONS: Dynamic optimized neighbor selection for smart blockchain networks. *Future Generation Computer Systems*, 130:75–90, 2022.

[86] Mishelle Cordero, Andrés Miniguano-Trujillo, Diego Recalde, Ramiro Torres, and Polo Vaca. Graph partitioning in connected components with minimum size constraints via mixed integer programming. 2022.

[87] Franz Rothlauf. *Design of modern heuristics: principles and application*, volume 8. Springer, 2011.

[88] Ilhem Boussaid, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information sciences*, 237:82–117, 2013.

[89] Hamza Baniata, Ahmad Anaqreh, and Attila Kertesz. Pf-bts: A privacy-aware fog-enhanced blockchain-assisted task scheduling. *Information Processing & Management*, 58(1):102393, 2021.

[90] Juan J Durillo, Hamid Mohammadi Fard, and Radu Prodan. Moheft: A multi-objective list-based method for workflow scheduling. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pages 185–192. IEEE, 2012.

[91] Amritraj Singh, Reza M Parizi, Meng Han, Ali Dehghantanha, Hadis Karimipour, and Kim-Kwang Raymond Choo. Public blockchains scalability: An examination of sharding and segregated witness. In *Blockchain Cybersecurity, Trust and Privacy*, pages 203–232. Springer, 2020.

[92] Shubin Cai, Ningsheng Yang, and Zhong Ming. A decentralized sharding service network framework with scalability. In *International Conference on Web Services*, pages 151–165. Springer, 2018.

[93] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30, 2016.

[94] Alex Manuskin, Michael Mirkin, and Ittay Eyal. Ostraka: Secure blockchain scaling by node sharding. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 397–406. IEEE, 2020.

[95] Vankamamidi S Naresh, VVL Divakar Allavarpu, Sivaranjani Reddi, Pilla Sita Rama Murty, NVS Lakshmipathi Raju, and RNV Jagan Mohan. A provably secure sharding based blockchain smart contract centric hierarchical group key agreement for large wireless ad-hoc networks. *Concurrency and Computation: Practice and Experience*, page e6553, 2022.

[96] Mengqian Zhang, Jichen Li, Zhaohua Chen, Hongyin Chen, and Xiaotie Deng. Cycledger: A scalable and secure parallel protocol for distributed ledger via sharding. In *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 358–367. IEEE, 2020.

[97] Chenyu Huang, Zeyu Wang, Huangxun Chen, Qiwei Hu, Qian Zhang, Wei Wang, and Xia Guan. Repchain: A reputation-based secure, fast, and high incentive blockchain system via sharding. *IEEE Internet of Things Journal*, 8(6):4291–4304, 2020.

[98] Yizhong Liu, Jianwei Liu, Qianhong Wu, Hui Yu, Hei Yiming, and Ziyu Zhou. Sshc: A secure and scalable hybrid consensus protocol for sharding blockchains with a formal security framework. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[99] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 931–948, 2018.

[100] Siddhartha Sen and Michael J Freedman. Commensal cuckoo: Secure group partitioning for large-scale services. *ACM SIGOPS Operating Systems Review*, 46(1):33–39, 2012.

[101] Gianmaria Del Monte, Diego Pennino, and Maurizio Pizzonia. Scaling blockchains without giving up decentralization and security: A solution to the blockchain scalability trilemma. In *Proceedings of the 3rd Workshop on Cryptocurrencies and Blockchains for Distributed Systems*, pages 71–76, 2020.

[102] P. Erdos and A. Renyi. On random graphs,. *Publicationes Mathematicae*, 6:290–297, 1959.