Applied Information Technology And Computer Science Vol. 5 No. 1 (2024) 016-033

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Comparative Study on Hash Function Algorithms for Blockchain Technology

Pun Kah Yien[1], Kamaruddin Malik[1]*, Sofia Najwa Ramli[1]

[1]Faculty Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2024.05.01.002
Received 18 May 2024; Accepted 22 May 2024; Available online 30 August 2024

Abstract*:* Cryptocurrency is the use of cryptography to create and distribute currency units making cryptocurrency a peer-to-peer digital exchange system. Bitcoin is a popular cryptocurrency that uses hash algorithms for its proof-of-work consensus mechanism. However, there are some specific attacks in Bitcoin regarding the hash function such as double spending and 51% attack. Therefore, this research paper is to make a comparative study of hash algorithms based on speed to increase the security of the Bitcoin blockchain. The comparison was made in Java programming by calculating the hashing speed to hash the test data of 1MB size in milliseconds with 100, 500, and 1000 loops repeated 10 times to get the average result using SHA256, KECCAK256, and BLAKE2b. In the 1000-loop experiment, KECCAK256 is approximately 33.29% faster than SHA256, while BLAKE2b is approximately 47.18% faster. According to the comparison analysis, BLAKE2b's hashing speed is the fastest among the selected hash algorithms. It could increase the security of the blockchain against 51% attack and double spending.

Keywords: Security, Hash, SHA256, KECCAK256, BLAKE2b

1. Introduction

The term cryptocurrency refers to a digital asset that circulates without the need for a central monetary authority (e.g., a government) [1]. Anyone can send and receive payments using this peer-to-peer system. Unlike physical money, cryptocurrency payments exist as digital entries in an online database that describe specific transactions and are not carried around and exchanged in the real world. Cryptocurrency funds can be transferred from one address to another via a public ledger and it will be recorded. Digital wallets are used to store cryptocurrency like our Touch N Go e-wallet stores our money.

With the growth of the cryptocurrency market, various types of cryptocurrencies are now appearing on the market that have entertainment value, such as Dogecoin, which is a cryptocurrency with high entertainment value. Consequently, those cryptocurrencies must be able to prove that their hash functions fulfill specific functional requirements such as security, decentralization, and transaction times. Nevertheless, it is approximated that there will be an attack known as a 51% attack and double spending on the market, which can disrupt the normal functioning of the market in a negative way.

*Corresponding author: malik@uthm.edu.my

There are various cryptocurrencies in the market, and each cryptocurrency uses various hash functions. Applying efficient hash functions to Bitcoin can enhance its security by mitigating the risks of 51% attacks and double spending. As well as being more secure, a higher hash speed enhances the network's security because a greater hash rate makes it more difficult to perform a successful 51% attack [2]. Thus, the difference in hash function used in Bitcoin needs to find out which one is more efficient and secure. Nonetheless, there have some problems with how we would determine whether they are viable to ensure that the data is willing to adopt them. The objectives of this research are to:

(i) to study the different hash algorithms for hashing in Bitcoin,
(ii) to design the programming code for computing hashing speed, and
(iii) to evaluate the efficiency of different hashing algorithms based on speed.

The rest of the paper is organized as follows: Section 2 is a Literature Review of the background of the research, Section 3 is the Methodology, Section 4 is the Result and Discussion while Section 5 is the Conclusion.

2. Literature Review

In this chapter, the knowledge related to the research is discussed. It included blockchain technology, cryptocurrency, attack on Bitcoin and the hash algorithms.

2.1 Blockchain Technology

There are many cryptographic techniques that are used in blockchain-based systems to maintain the integrity of ledgers. In a blockchain, transactions are stored in a physical chain of fixed-length blocks. Each block contains 1 to N transactions, which are validated before they are inserted into the block. Each completed block is added to the chain at the end. The block data cannot be altered retroactively once it has been recorded; the blocks are designed to resist modification once recorded. Blockchain databases can be managed autonomously using peer-to-peer networks and distributed timestamping servers. A blockchain is simply a distributed ledger with an open and verifiable record of transactions between two parties that can be used to record and verify transactions efficiently and permanently [3].

2.2 Cryptocurrency

The use of cryptography to create and distribute currency units makes cryptocurrency a peer-to-peer digital exchange system. There is no central authority involved in this process as transactions are distributed for verification. Cryptocurrency makes use of cryptographic methods to ensure transactions are valid and unique. Cryptocurrencies transmit digital information using digital methods [4]. Essentially, cryptocurrency work under blockchains functions as public ledgers shared by participants, with rewards given to those who run the transaction networks [5]. The word cryptocurrency refers to a method of securing transactions and creating new units of money that are controlled by cryptography.

2.2.1 Bitcoin

Hash functions used in Bitcoin blockchain technology need to be computationally efficient to ensure the smooth functioning of the blockchain network. Hash functions that are computationally inefficient can slow down the network and increase the time and cost of mining [6]. Therefore, optimizing the efficiency of hash function implementation is important for Bitcoin mining [7].

2.2.2 Ethereum

A generalized blockchain platform was envisioned with the Ethereum platform in November 2013 when it combined the notion of public economic consensus (or proof of work or stake) with the abstraction power of a Turing-complete virtual machine to create a more generalized blockchain platform [8]. Accounts are Ethereum's basic unit. Anyone who wants to send transactions to the blockchain needs an account. There are two kinds of Ethereum accounts: Externally Owned Accounts (EOA), which enables

users to send transactions through them, and Contract Accounts, which allow users to send internal transactions. The account also acts as two keys which are a private key and a public key. Every transaction requires the account address, which is derived from the last 20 bytes of a public key [9].

2.3 Bitcoin

The popularity of Bitcoin has increased since 2008 when it was introduced as an electronic currency. The bitcoin system stores transactions in a public ledger ('the blockchain') on a peer-to-peer network, which is decentralized. The Bitcoin network allows for the issuance of decentralized currency and the clearance of transactions. As part of the security of the blockchain, bitcoin mining uses a compute-intensive algorithm to maintain the integrity of the transactions such as preventing double spending of bitcoins and the tampering of confirmed transactions [10].

2.3.1 Bitcoin’s Proof of Work (PoW)

Miners and mining are the terms associated with the nodes that calculate the Hashes as well as the process of calculating the Proof of Work. PoW is used in Bitcoin mining [11]. For the authentication of PoW, a complicated computation is required. The algorithm that is being used for proof of work is called Hashcash. Miners in Hashcash must find a nonce, which, when combined with the previous block's hash, will produce a hash that has a specified number of zeros on its front [12]. It is difficult to mine a block as its SHA-256 hash must be lower than or equal to the target hash of the block's header in order to be accepted as a block by the network. Each Bitcoin client shares a 256-bit integer called a target; a lower target indicates a greater difficulty [13].

2.3.2 Mining

The mining process will validate and add all transactions to the public ledger. It is the miner's job to determine whether the currency belongs to the payer or if the payer is trying to double-spend when a new transaction occurs. In Blockchain, it is possible to see who owns the currency and who is the holder. The resource-intensive task may be Proof of Work (PoW) or Proof of State (PoS) and many other mining techniques depending on the cryptocurrency that they have applied to it. Mining requires resource-intensive tasks, and miners produce evidence that the work has been completed. By completing this task, the miners will be unable to construct false identities and manipulate them in any way. After that, verification is performed on the proof to ensure that the task has been completed. Afterward, the miner validates the validity of each transaction in the block, and the block is recorded in the blockchain if all the transactions are valid [13].

2.3.3 Blockchain in Bitcoin

When it comes to blockchain, it's important to mention transactions. In order to prove the chronological order of transactions, Satoshi Nakamoto proposed a P2P distributed timestamp server system[11]. Digital signatures are a key component of electronic coins such as Bitcoin. The transaction is described as consisting of the public key of the next owner and the set of a digitally signed hash of the last transaction. The transaction must be signed with the private key, and it must be verified with the public key [14].

2.4 Attack on Bitcoin

White hat hackers found 43 security flaws in different blockchain and cryptocurrency systems in less than a month's time in March 2019. Popular blockchain projects including EOS, Coinbase, and Tezos have vulnerabilities that researchers have discovered [15]. As a demonstration of cybercriminal intelligence, cybercriminals target large networks such as Bitcoin and Ethereum to commit fraud and theft. There are some specific attacks in Bitcoin are blockchain network attacks, user wallet attacks, transaction verification mechanism attacks, and mining attacks.

2.4.1 51% Attack

It is a technique in which an attacker can gain 51% of the hashing power over a network when they possess 51% of the hashing power. A private chain of blocks is created as the first step in this attack, which is completely isolated from the real version of the chain in which it is launched. There will be a subsequent step where the isolated chain will be presented to the network to establish itself as one of the real chains [16]. In accordance with the longest chain rule, the blockchain policy follows this principle [11]. Figure 1 shows the process of a 51% attack on a blockchain.

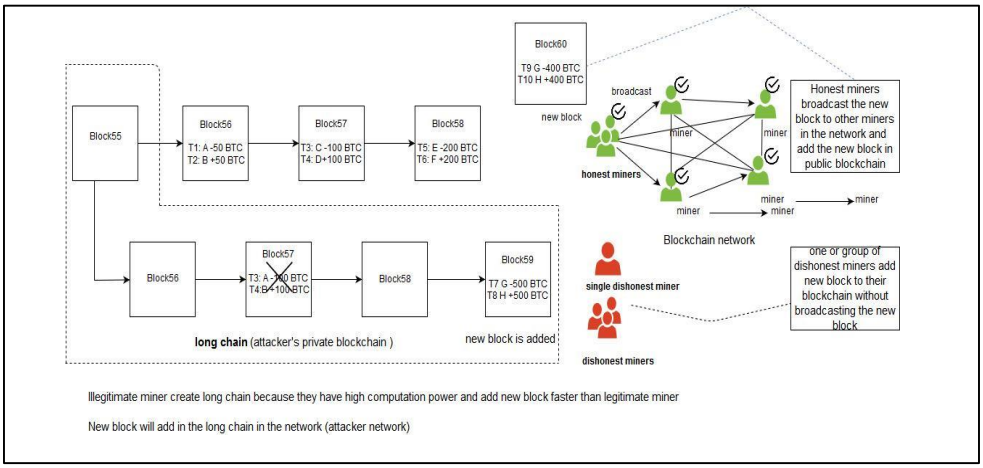

Figure 1: 51% attack [17]

As the hash speed of the algorithm increases, it can help to a certain extent prevent a 51% attack in the blockchain, but it still isn't a foolproof solution to prevent 51% attacks. It is valuable for a mining network to have a higher hash rate distributed among honest miners, as a higher hash rate inhibits an attacker from gaining majority control over the network and executing malicious activities such as double spending or 51% attacks during an attack[16].

2.4.2 Double Spending

The term "double spending" refers to the use of the same cryptocurrency more than once by the same consumer. In the case of digital currencies, the most common security issue is the possibility that a token can be spent more than once, which is known as double spending [18]. An attacker is able to implement this kind of attack with relative ease on a blockchain based on proof-of-work, as the attacker can exploit the intermediate period between the initiation and confirmation of two transactions to initiate an attack quickly. Since the attacker has already obtained the output of the first transaction before it is mined as invalid, double spending will result if the second transaction is mined as invalid [19]. Figure 2 shows the process of a double spending attack in a blockchain.

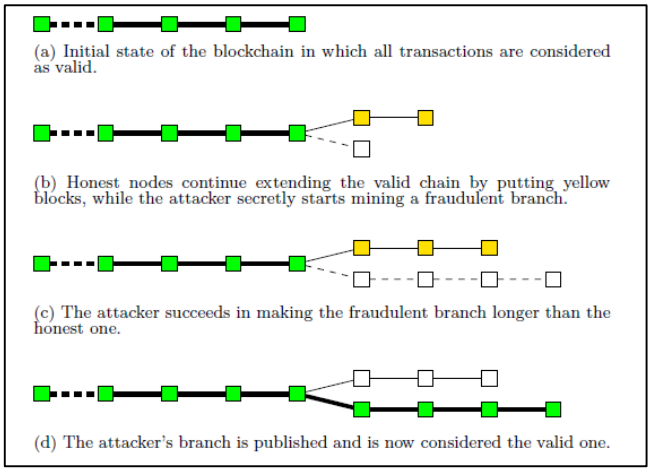

Figure 2: Double Spend Attack [20]

2.5 Metric of Hash Function

Among blockchain technology's key metrics is efficiency, particularly when considering hash functions. Blockchain networks are susceptible to security and performance issues due to low computational efficiency of hash functions [21]. In order to determine the level of performance of hash algorithms, they need to be thoroughly checked, and a comparative analysis can be conducted to determine if they have the same level of performance based on the speed of operation, security concerns, and efficiency of each algorithm [22].

2.5.1 Speed

The speed or throughput of an algorithm is often used as a measure of the efficiency of the algorithm. The use of faster algorithms is advantageous for applications that need high-speed hashing, such as cryptographic operations, and data integrity checks, where high-speed hashing is required to ensure data integrity[23]. For hash algorithms, the average speed of hashing a message can be an efficient metric based on speed. Meanwhile, there are some ways of measuring speed, including cycles/bytes or the duration of time it takes to hash a message [24]. The speed of hash algorithms can prevent attacks by computing more hashes in a certain, hence increasing the hash rate on the blockchain network. The higher the hash rate, the higher hash rate makes it more challenging for an attacker to overpower the network and execute a successful 51% attack.

2.5.2 Computational Complexity

Stratification and comparison of computational resources required to solve problems are the objectives of computational complexity theory, a subfield of theoretical computer science. Processor cycles and memory space are typically measured in terms of how much memory space is required to return a solution when measuring the resources involved in executing an algorithm to solve an instance of a problem.[25]. Algorithms with higher efficiency have lower computational complexity, which means that they can run on a wide range of devices [26].

2.5.3 Collision Resistance

Hash algorithms aim to create unique hash values for different inputs in order to minimize the likelihood of collisions between two different inputs resulting in the same hash value being produced by two different hash algorithms. In fact, when using a more efficient algorithm, there is less likelihood that a collision will occur, which will result in fewer additional checks or rehashes being needed to handle collisions [27].

2.6 Hash Function

Hash functions have been used in computer science and they refer to a function that compresses a string of arbitrary input into a string of a fixed length after it has been compressed by the hash function. Hash functions play an important role in cryptography and can be used to achieve a range of security goals, including authenticity, digital signatures, pseudo-numerical generation, digital steganography, and digital time stamp [28]. There are 3 types of hash functions were chosen to make comparisons based on the speed which are SHA256, KECCAK256, and BLAKE2b. SHA256 is the original hash function used by Bitcoin, therefore it must be chosen of one of the hash functions to be compared. Next, the SHA-3 project was started by NIST in 2006, and Keccak was selected as the winner of the competition on October 2, 2012 [29]. The KECCAK256 algorithm is well known for its high performance and speed when it comes to computation speed, particularly on 64-bit platforms[30]. The hash algorithm has been optimized to maximize its performance and is able to process large amounts of data at a very fast speed, which makes it ideal for applications such as block validation, transaction processing, and mining, among others. BLAKE2b is designed to operate on 64-bit platforms, allowing Blake2b to take advantage of 64-bit arithmetic, 64-bit registers, and 64-bit parallel computations on 64-bit words. Since it can compute hashes faster than any other hash function, it is a particularly suitable choice for blockchain implementations running on 64-bit architectures compared to other hash functions because it can be used to accelerate hash computations [31].

2.6.1 SHA-256

Cryptographic hashes are algorithms that process arbitrary (usually very long) strings of input into short fixed-length output strings. Prior to very recently, cryptanalysis of cryptographic hash functions had made very slow progress. Cryptographic researchers [32] have been surprised by how fast cryptanalysis of hash functions has progressed, including attacks on MD5 that detect collisions, attacks on SHA-0[33] that detect multi-collision, and attacks on MD5. Since SHA-256 has a more complex message schedule and round function, these techniques do not apply. As a new generation of SHA functions, SHA-256 was proposed in 2000 and adopted as a FIPS standard in 2002 [34].

SHA-256 compresses data in 64 rounds. It consists of two linear functions Σ0, Σ1 along with two non-linear functions, CH, and MJ, as well as an arithmetic addition function and a round-dependent constant[35]. Figure 3 shows the round function for SHA-256.

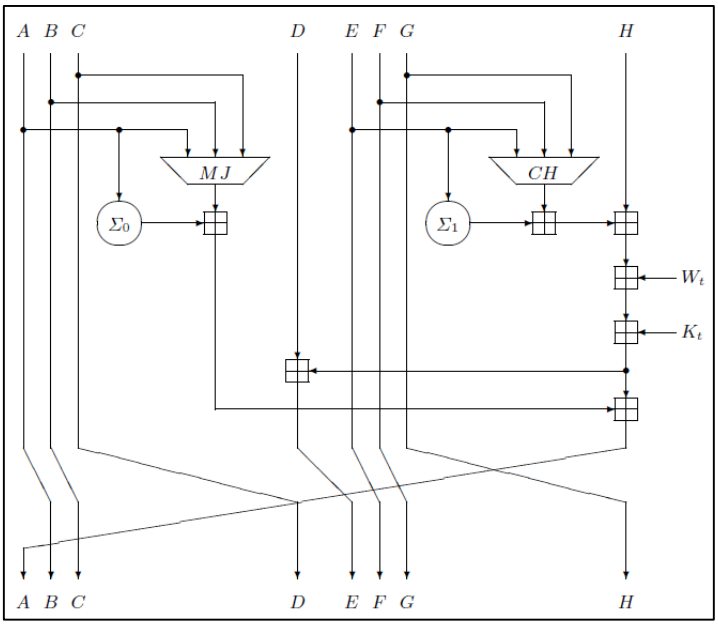

Figure 3: Round Function For SHA-256 [35]

Which X is X's bitwise complement. With the word Wt and the constant Ki as input, the t-th round of the compression function updates the 8 registers. Following is the algorithm used by the compression function to update the 8 registers.

2.6.2 Keccak-256

The Keccak hash function is based on sponge construction [36]. In message M, the bits are padded and separated into blocks of r bits each. Following an initial value (IV), the first r bits of the b-bit state are XORed with the message block, and then the permutation f is applied. All message blocks are processed in this manner until they are all processed. Once the first r bits are obtained, f is applied again until all required digest bits are obtained, and the process is repeated until all required digest bits are obtained. Based on the number of digest bits λ requested, the number of iterations is determined. At the end of the output process, the output is truncated to its first λ bits [37]. Figure 4 shows the diagram of the sponge function.

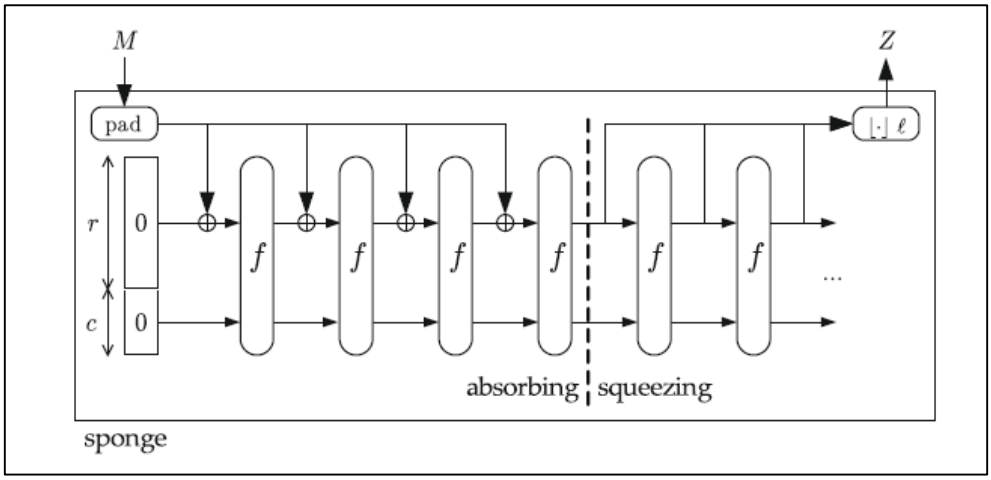

Figure 4: The sponge function [37]

Keccak hash functions are defined by the designers by specifying the parameters set of (r, c, l) as well as the underlying permutation f. The IV is set to be all "0"s. Among the seven Keccak-f permutations, Keccak-f [b] is chosen from the set of seven Keccak-f permutations, where b∈ {25, 50, 100, 200, 400, 800, 1600}. The default Keccak-f version has a size of b = 1600 bits, which can be represented with five times five 64-bit lanes denoted as A[x, y] where x is column index, while y is row index. The x and y indexes are from the set [0, 1, 2, 3, 4] and work mod 5 without further specification.

2.6.3 BLAKE2

BLAKE2b and BLAKE2s are the two main algorithms in the BLAKE2 family. BLAKES2b produces digests of any size between one and 64 bytes and is optimized for 64-bit platforms, including NEON-compatible ARMs. It produces digests of any size between 1 and 32 bytes and is optimized for 8- to 32-bit platforms. [31]. When used on the CPU size for which it is optimized, the BLAKE2s will be up to twice as fast as the BLAKE2b; for example, BLAKE2s should be twice as fast on a Tegra 2 (32-bit ARMv7-based SoC), while BLAKE2b should be more than 1.5 times faster than BLAKE2s on an AMD A10-5800K (64-bit, Piledriver microarchitecture) [31]. BLAKE2b generates digests of any length between 1 and 64 bytes, which may be a limitation depending on the software being used[38]. Figure 5 shows the basic round of BLAKE2b.

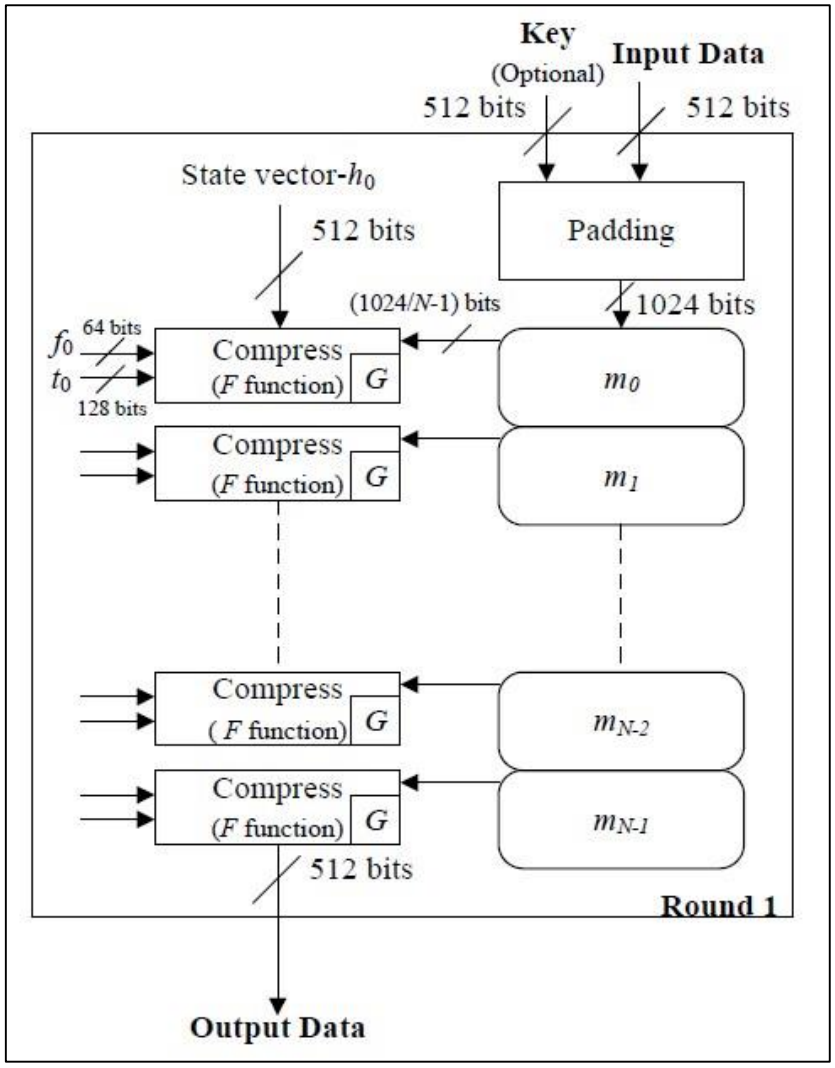

Figure 5: The basic round of BLAKE2b is shown in the block diagram [39]

2.7 Research Justification

The speed of the hash function is an important metric for blockchain security [40]. The hash rate in blockchains can be affected by the hash function speed. The hash rate is a metric that quantifies the computational power present within a blockchain network. It is determined by the number of computational guesses made per second. This collective hash rate plays a crucial role in assessing the security and mining complexity of the blockchain network[41]. Thus, the hash rate is important because the security in blockchain technology is determined by the total hash rate of the transactions that are recorded in the blockchain [42]. Therefore, a high hash rate improves security, and this is its primary advantage. Hash rate can be considered as a measure of computer power contributing to the network. This means that the hash algorithms used in the blockchain can have a significant impact on the overall hash rate [43]. Cryptographic hashes are important because it used to hash of previous blocks, timestamps, and transaction information are all part of every blockchain security mechanism. So, it is essential that a good hash function has two main properties: it should be able to calculate quickly and minimally collide with itself [44]. It was found out to be true however that most of the hashes were either usable or inefficient when it comes to speed [22].

Table 1: Research Gap

Author	Hash Algorithms Used	Description	Year
[45]	MD5, RIPEMD-160, Secure Hash Algorithm (SHA)	This paper provides an overview of these common algorithms and focuses on how vulnerable they are to common attacks. The time inefficiency and collision attacks that these common hash function algorithms experience.	2018
[46]	SHA-2, SHA-3-E, SHA-3-S, BALKE2	The evaluation results of various Ethereum versions with BLAKE2 and SHA-2 that using a more efficient hash function could significantly enhance the performance of a private blockchain.	2020
[22]	MD5, SHA-1, SHA-256, SHA-512	A comparison is provided between the SHA family and MD5 based on their performance, their security concerns, and the use of the Secure Hash Algorithm.	2021
[47]	SHA-256, BLAKE2b	BLAKE2b hashes a single plaintext faster than SHA-256, but SHA-256 operates much more quickly on complex architectures like PoW.	2022
This paper	SHA256, KECCAK256, BLAKE2b	A comparative study on hash algorithms based on speed to improve security by preventing 51% attack and double spending on Bitcoin.	Current

3. Methodology

This section is about the methodology with the research framework to conduct the research.

3.1 Research Framework

The first step is to study the different hash algorithms. The second step is to choose algorithms to compare. The third is to choose evaluations parameter for the hash algorithms. The fourth step is to implement the code for the comparison experiment. Next is run comparison experiments on chosen hash algorithms based on speed evaluation parameters using the prepared codes. The last step is to record and analyse the result. The research framework is shown in Figure 6.

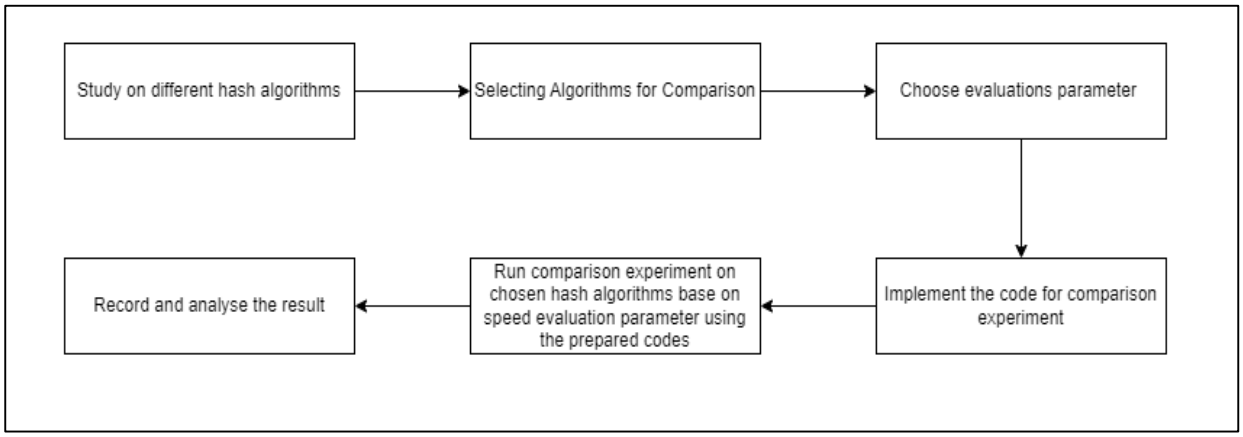

Figure 6: Research framework

3.2 Study on Different Hash Algorithms

The process involved several key steps, beginning with an extensive literature review to identify a diverse set of hash algorithms. Hash algorithms are primarily used for data integrity, which is a component of the CIA triad. Research papers, articles, and relevant scholarly sources were carefully analysed to understand the theoretical foundations, practical implementations, and performance characteristics of each algorithm. Additionally, open-source implementations of these algorithms were examined to assess their availability and ease of integration within the project framework.

3.3 Selecting Algorithms for Comparison

The selection of appropriate algorithms is crucial to ensure a comprehensive evaluation of different approaches in addressing the problem statement. A number of factors were taken into account, including algorithm popularity, relevance to the project domain, and potential impact. To achieve this, an extensive literature review was conducted, where relevant research papers, articles, and existing implementations were analysed.

3.4 Choose Evaluation Parameters

The evaluation parameters were carefully selected to provide a comprehensive and objective analysis of the comparison and to measure its alignment with the research objectives. The chosen parameters are to improve Bitcoin and prevent some attacks such as 51% attack and double spending. Computational efficiency evaluates the solution's speed and resource utilization. The evaluation parameter unit is in milliseconds of speed in hashing.

3.5 Implementing Code for Comparison Experiment

The implementation process involved translating the algorithmic steps or methodologies into executable code, including appropriate algorithms, and any necessary libraries or dependencies.

```
import org.bouncycastle.crypto.digests.*;
```

Figure 7: Import org.bouncycastle.crypto.digests.*

Figure 7 shows the org.bouncycastle.crypto.digests package is a part of the Bouncy Castle cryptographic library, which is widely used for implementing cryptographic operations in Java applications. Within this package, the digests subpackage specifically focuses on providing various digest algorithms that need to compare for this research, which are cryptographic hash functions used for generating fixed-size hash values from input data of arbitrary size.

```
// Initialize the SHA-256 digest
SHA256Digest sha256Digest = new SHA256Digest();
byte[] sha256Buffer = new byte[4096];
```

Figure 8: Code for initializing the SHA256 digest

Figure 8 explain an implementation of the SHA-256 hash function provided by the 'SHA256Digest' class. This hash function produces a 256-bit hash value. Then Initiate the digest by calling 'SHA256Digest sha256Digest = new SHA256Digest();'.

```
// Initialize the Keccak-256 digest
KeccakDigest keccakDigest = new KeccakDigest(256);
byte[] keccakBuffer = new byte[4096];
```

Figure 9: Code for initializing the KECCAK256 digest

Based on Figure 9, KECCAK256 hash function is provided by the 'keccakDigest' class. This hash function produces a 256-bit value. Then Initiate the digest by calling 'KeccakDigest keccakDigest = new KeccakDigest(256);'.

```
// Initialize the Blake2b digest
Blake2bDigest blake2bDigest = new Blake2bDigest();
byte[] blake2bBuffer = new byte[4096];
```

Figure 10: Code of initializing the BLAKE2b digest

Based on Figure 10, the implementation of the BLAKE2b hash function is provided by the 'Blake2bDigest' class. This hash function produces a 256-bit hash value. Then Initiate the digest by calling Blake2bDigest blake2bDigest = new Blake2bDigest();'.

```
44              // Hash the file using SHA-256
45              long startTime = System.currentTimeMillis();
46              try (InputStream inputStream = new FileInputStream(inputFile)) {
47                  int bytesRead;
48                  while ((bytesRead = inputStream.read(sha256Buffer)) != -1) {
49                      sha256Digest.update(sha256Buffer, 0, bytesRead);
50                  }
51              }
52              long sha256TimeInMs = System.currentTimeMillis() - startTime;
53
54              // Get the SHA-256 hash value
55              sha256Hash = new byte[sha256Digest.getDigestSize()];
56              sha256Digest.doFinal(sha256Hash, 0);
```

Figure 11: Partial code for hashing and get the hash value

Based on Figure 11, the 'update' method in line 49 is used to feed data to the hash function, and the 'doFinal' method in line 56 is called to retrieve the final hash value.

```
long startTime = System.currentTimeMillis();
```

Figure 12: Code for start record time

Based on Figure 12 the value returned by 'System.currentTimeMillis()' is stored in the variable 'startTime'. This marks the starting point or reference time for the subsequent operation and is used to capture the current time in milliseconds.

```
long sha256TimeInMs = System.currentTimeMillis() - startTime;
```

Figure 13: Code calculates the duration

The value of 'System.currentTimeMillis()' at the end of the operation is subtracted from the value stored in the 'startTime' variable, which represents the starting time of the operation. The result of this subtraction represents the time elapsed during the operation. The resulting time difference is stored in the variable 'sha256TimeInMs', which represents the elapsed time of the SHA-256 hashing operation.

3.6 Comparison Experiment on Hash Algorithms based on Speed Evaluation Parameter

This section describes the methodology employed to conduct a comparison experiment on the chosen hash algorithms, focusing on the speed evaluation parameter, using the prepared codes in this research. The system that will run this program is 64 bits Windows with IntelI Core I i7-1065G7 CPU @ 1.30GHz, 1498 Mhz, 4 Core(s), 8 Logical Processor(s), and 8GB RAM. To conduct this, a controlled experimental setup was created in Java language, ensuring consistent testing conditions across all algorithms. The prepared codes were implemented to perform hashing operations on fixed input sizes and simulate the blockchain size which is 1Mb of input. The execution time of each algorithm was recorded multiple times. To ensure statistical significance, the experiment was repeated 10 times with 100, 500, and 1000 loops to obtain reliable and consistent results.

3.7 Record and Analyse the Result

During the experimental phase, the measurements and observations of the relevant parameters were meticulously recorded and stored in a structured format, ensuring the integrity and traceability of the

data. The recorded data was then subjected to a comprehensive analysis, employing appropriate statistical techniques and tools. This analysis involved applying measures such as means.

3.8 Performance Metric – Speed

The performance metric speed is measured by the time taken to complete the work which is the time taken to hash 1Mb of test data as shown in Equation 1. The program starts recording the time when starts hashing the test data and ends at the time when finishing hashing the test data. Then the average is the time accumulated from each time taken with the total and divided by the number of iterations or loops as in Equation 2.

$$Speed: Input = Start\ Time\ of\ Hashing - End\ Time\ of\ Hashing \quad \text{Eq1}$$

$$Average = Total\ Time\ of\ Hashing\ /\ Number\ of\ Loops \quad \text{Eq2}$$

4. Result and Discussion

This section discussed the performance of the hashing speed of selected hash algorithms and the analysis of the experiments.

4.1 Performance of Hashing Speed

The experiment aimed to investigate the hashing time of a 1MB JSON file using the chosen hashing algorithms: SHA256, KECCAK256, and BLAKE2b. To simulate this, the experiments ran each hash function on the JSON file and recorded the execution time for each one. The experiments are done by calculating the hash value of the test data repeatedly 10 times in all loops of 100 in Experiment 1, 500 in Experiment 2, and 1000 in Experiment 3. The calculation taken is then recorded in milliseconds. These values were then used to calculate the average time, which was obtained by computing the mean of the recorded times. The results of the experiment are presented in the table below, which shows the average hashing time in milliseconds.

4.1.1 Experiment 1: Compare hashing speed on test data for 100 loops.

Based on Table 2, the BLAKE2b algorithm demonstrates the fastest average hashing time of 4.28 milliseconds, followed by KECCAK256 with an average hashing time of 4.97 milliseconds. SHA256 is the slowest among the three algorithms, with an average hashing time of 7.41 milliseconds. The result shows that BLAKE2b can provide a higher hash rate because it can hash data using less time which means it can complete a hash in the shortest time among the hashing algorithms.

Table 2: Time taken to hash the test data using SHA256, KECCAK256, and BLAKE2b for 100 loops

Experiments (100 loops)	Hashing Time (milliseconds) Average		
	SHA256	Keccak256	BLAKE2b
1	7.41	4.84	4.11
2	7.02	4.64	4.15
3	7.48	5.12	4.40
4	7.24	5.02	4.24
5	7.51	5.14	4.51
6	7.47	4.84	4.24
7	7.39	5.04	4.28
8	7.78	5.10	4.34
9	7.34	4.98	4.28
10	7.47	5.01	4.29
Average	7.41	4.97	4.28

4.1.2 Experiment 2: Compare hashing speed on test data for 500 loops.

Based on Table 3, the BLAKE2b algorithm continues to demonstrate the fastest average hashing time of 3.82 milliseconds. SHA256 follows with an average hashing time of 7.05 milliseconds, and KECCAK256 lags slightly behind with an average hashing time of 4.77 milliseconds. These findings emphasize the efficient performance of BLAKE2b and highlight its suitability for scenarios where fast hashing is crucial. The results show that BLAKE2b can provide a higher hash rate because it can hash data in less time, completing a hash in the shortest time of the hashing algorithms.

Table 3: Time taken to hash the test data using SHA256, KECCAK256, and BLAKE2b for 500 loops

Experiments (500 loops)	Hashing Time (milliseconds) Average		
	SHA256	KECCAK256	BLAKE2b
1	7.40	5.01	4.03
2	6.91	4.65	3.78
3	6.74	4.55	3.57
4	6.77	4.55	3.64
5	7.89	5.34	4.36
6	7.28	4.89	3.97
7	6.94	4.73	3.80
8	6.84	4.63	3.63
9	6.72	4.62	3.64
10	7.02	4.75	3.82
Average	7.05	4.77	3.82

4.1.3 Experiment 3: Compare hashing speed on test data for 1000 loops.

Based on Table 4, the BLAKE2b algorithm continues to demonstrate the fastest average hashing time of 3.65 milliseconds. KECCAK256 follows with an average hashing time of 4.61 milliseconds, and SHA256 lags slightly behind with an average hashing time of 6.91 milliseconds. These findings reaffirm the better performance of BLAKE2b and its suitability for scenarios where fast hashing is crucial. The result shows that BLAKE2b can provide a higher hash rate because it can hash data using less time which means it can complete a hash in the shortest time among the hashing algorithms.

Table 4: Time taken to hash the test data using SHA256, KECCAK256, and BLAKE2b for 1000 loops

Experiments (1000 loops)	Hashing Time (milliseconds) Average		
	SHA256	KECCAK256	BLAKE2b
1	6.93	4.63	3.63
2	6.90	4.59	3.69
3	7.15	4.83	3.85
4	6.87	4.59	3.59
5	6.81	4.57	3.6
6	6.82	4.52	3.59
7	6.80	4.55	3.60
8	6.94	4.58	3.67
9	6.89	4.60	3.68
10	7.00	4.68	3.66
Average	6.91	4.61	3.65

4.3 Experiment Analysis

In all three experiments, the average hashing times vary for each algorithm, but their ranking in terms of hashing speed remains consistent. SHA256 consistently has the longest average hashing time, while KECCAK256 consistently performs better than SHA256, and BLAKE2b consistently demonstrates the

fastest hashing speed among the three algorithms. The results highlight the reliable performance order of the algorithms in terms of speed. Regardless of the specific average hashing times observed in each experiment, it is evident that SHA256 is the slowest, KECCAK256 is faster than SHA256, and BLAKE2b consistently outperforms both SHA256 and KECCAK256 in terms of hashing speed. As a result of the experiments, the BLAKE2b algorithm can prevent Bitcoin attacks and ensure a better level of security. By using the BLAKE2b algorithm, which consistently demonstrates the fastest hashing speed among the tested algorithms, the blockchain network can effectively increase its overall hash rate. A higher hash rate makes it more challenging for an attacker to overpower the network and execute a successful 51% attack. In this context, the faster hashing speed of BLAKE2b enhances the security of the system, mitigating the risk of a 51% attack and ensuring the integrity and reliability of the blockchain.

Table 4.3: Experiments of average time taken to hash the test data using SHA256, KECCAK256, and BLAKE2b

Experiments (loops)	Hashing Time (milliseconds) Average		
	SHA256	KECCAK256	BLAKE2b
100	7.41	4.97	4.28
500	7.05	4.77	3.82
1000	6.91	4.61	3.65

In overall experiments, when more hash calculations are performed, the better performance comparison results can be achieved. In other words, the bigger the test data used in the hashing speed comparison experiments, it shows that these three hash algorithms give a better performance. For example, the experiment for 100 loops with BLAKE2b was 4.28 milliseconds, the experiment for 500 loops was 3.82 milliseconds and the experiment for 1000 loops was 3.65 milliseconds. The time taken for 500 loops is approximately 10.75% faster than the time taken for 100 loops. While the time taken for 1000 loops is approximately 4.45% faster than the time taken for 500 loops.

5. Conclusion

The objective of this research is to study the related metric for hashing in Bitcoin, to design the programming code for computing hashing speed, and to compare the efficiency of different hashing algorithms based on speed.

In the 100-loop experiment, compared to SHA256, KECCAK256 is approximately 32.93% faster, while BLAKE2b is approximately 42.24% faster. This indicates that both KECCAK256 and BLAKE2b outperform SHA256 significantly in terms of speed.

In the 500-loop experiment, compared to SHA256, KECCAK256 is approximately 32.24% faster, while BLAKE2b is approximately 45.81% faster. These percentages highlight the substantial speed advantage of both KECCAK256 and BLAKE2b over SHA256.

In the 1000-loop experiment, KECCAK256 is approximately 33.29% faster than SHA256, while BLAKE2b is approximately 47.18% faster. These percentages reinforce the consistent speed superiority of both KECCAK256 and BLAKE2b over SHA256.

In conclusion, the comparative result shows that BLAKE2b is faster than the hash algorithm of KECCAK256 and the original hash algorithm used by Bitcoin which is SHA256. Therefore, in terms of the speed of a hash algorithm, BLAKE2b is suitable for Bitcoin as one of the security mechanisms to improve the security of the blockchain as explained in Section 2.

5.1 Limitation of Research

The limitation of the research is the study mainly focuses on comparing hashing algorithms based on speed. Even though speed is a significant metric, efficiency, and security are not solely determined by

it. There are also several other factors that contribute to the success of the Bitcoin network, including memory usage, resistance to attacks, and compatibility with the Bitcoin network. Therefore, the comparative study of hashing algorithms may be incomplete if these factors are ignored.

Moreover, despite these objectives, none of them explicitly take into account real-life factors, such as network conditions, mining difficulty, or other factors that might impact network participation. Hashing algorithms' performance and efficiency can be affected significantly by these factors.

5.2 Recommendation Future Work

There is some future work for this research. Besides speed, Bitcoin hashing also provides other important metrics. It would be useful to study and analyse memory usage, energy consumption, and collision resistance in future work and it would be more comprehensive to evaluate hashing algorithms in Bitcoin considering these metrics and their impact on efficiency and security.

Next, for a better understanding of how hashing algorithms perform in cryptocurrency, future work should incorporate more realistic scenarios. To gain valuable insight into the efficiency and effectiveness of Bitcoin hashing algorithms, data from mining pools and actual mine operations should be analysed in real-world conditions. This could involve collaboration with mining pools or access to mining hardware to gather data on hashing speed, energy consumption, and other relevant metrics.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support throughout the process of conducting this project

References

[1] Voigt and Rosen, “What is cryptocurrency and how does it work?,” *Forbes Advisor INDIA*, Jan. 11, 2018.

[2] M. Das, H. Luo, and J. C. P. Cheng, “Securing interim payments in construction projects through a blockchain-based framework,” *Autom Constr*, vol. 118, Oct. 2020, doi: 10.1016/j.autcon.2020.103284.

[3] J. J. Bambara and P. R. Allen, “Introduction to Blockchain,” in *Blockchain : a practical guide to developing business, law, and technology solutions*, New York: Mc Graw Hill Education, 2018, pp. 1–32.

[4] S. Scholarlycommons and R. Farell, “An Analysis of the Cryptocurrency Industry,” 2015. [Online]. Available: https://repository.upenn.edu/wharton_research_scholars/130

[5] A. Elbahrawy, L. Alessandretti, A. Kandler, R. Pastor-Satorras, and A. Baronchelli, “Evolutionary dynamics of the cryptocurrency market,” *R Soc Open Sci*, vol. 4, no. 11, Nov. 2017, doi: 10.1098/rsos.170623.

[6] J. Fu, S. Qiao, Y. Huang, X. Si, B. Li, and C. Yuan, “A Study on the Optimization of Blockchain Hashing Algorithm Based on PRCA,” *Security and Communication Networks*, vol. 2020, 2020, doi: 10.1155/2020/8876317.

[7] X. Zhang and H. Hu, “Optimization of Hash Function Implementation for Bitcoin Mining,” 2019.

[8] V. Buterin, “Ethereum: Platform Review Opportunities and Challenges for Private and Consortium Blockchains,” 2020.

[9] Sara Rouhani and Ralph Deters, "Performance Analysis of Ethereum Transactions in Private Blockchain," 2017.

[10] H. Vranken, "Sustainability of bitcoin and blockchains," *Current Opinion in Environmental Sustainability*, vol. 28. Elsevier B.V., pp. 1–9, Oct. 01, 2017. doi: 10.1016/j.cosust.2017.04.011.

[11] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: www.bitcoin.org

[12] A. Back, "Hashcash-A Denial of Service Counter-Measure," 2002.

[13] U. Mukhopadhyay, A. Skjellum, O. Hambolu, J. Oakley, L. Yu, and R. Brooks, "A Brief Survey of Cryptocurrency Systems," 2016.

[14] D. Vujičić, D. Jagodić, and S. Ranđić, "Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview," 2018.

[15] M. Binti Malik and M. Fadli Bin Zolkipli, "Blockchain Threats: A Look into the Most Common Forms of Cryptocurrency Attacks," 2023. [Online]. Available: www.majmuah.com

[16] S. Sayeed and H. Marco-Gisbert, "Assessing blockchain consensus and security mechanisms against the 51% attack," *Applied Sciences (Switzerland)*, vol. 9, no. 9, May 2019, doi: 10.3390/app9091788.

[17] Anitan and Vijayalakshmim, "Blockchain Security Attack: A Brief Survey," 2019.

[18] C. Pérez-Solà, S. Delgado-Segura, G. Navarro-Arribas, and J. Herrera-Joancomartí, "Double-spending prevention for Bitcoin zero-confirmation transactions," *Int J Inf Secur*, vol. 18, no. 4, pp. 451–463, Aug. 2019, doi: 10.1007/s10207-018-0422-4.

[19] X. Li, P. Jiang, T. Chen, X. Luo, and Q. Wen, "A survey on the security of blockchain systems," *Future Generation Computer Systems*, vol. 107, pp. 841–853, Jun. 2020, doi: 10.1016/j.future.2017.08.020.

[20] C. Pinzón and C. Rocha, "Double-spend Attack Models with Time Advantange for Bitcoin," *Electron Notes Theor Comput Sci*, vol. 329, pp. 79–103, Dec. 2016, doi: 10.1016/j.entcs.2016.12.006.

[21] J. Fu, S. Qiao, Y. Huang, X. Si, B. Li, and C. Yuan, "A Study on the Optimization of Blockchain Hashing Algorithm Based on PRCA," *Security and Communication Networks*, vol. 2020, 2020, doi: 10.1155/2020/8876317.

[22] M. Parmar and H. J. Kaur, "Comparative Analysis of Secured Hash Algorithms for Blockchain Technology and Internet of Things," 2021. [Online]. Available: www.ijacsa.thesai.org

[23] T. Newe, M. Rao, D. Toal, G. Dooly, E. Omerdic, and A. Mathur, "Efficient and high speed FPGA Bump in the wire implementation for data integrity and confidentiality services in the IoT," in *Smart Sensors, Measurement and Instrumentation*, Springer International Publishing, 2017, pp. 259–285. doi: 10.1007/978-3-319-47319-2_13.

[24] Y. Wang, L. Chen, X. Wang, G. Wu, K. Yu, and T. Lu, "The design of keyed hash function based on CNN-MD structure," *Chaos Solitons Fractals*, vol. 152, Nov. 2021, doi: 10.1016/j.chaos.2021.111443.

[25] Dean and Walter, "Computational Complexity Theory," in *The {Stanford} Encyclopedia of Philosophy*, {F}all 2021.Metaphysics Research Lab, Stanford University, 2021.

[26] J. Karásek, R. Burget, and O. Morský, "Towards an automatic design of non-cryptographic hash function," in *2011 34th International Conference on Telecommunications and Signal Processing, TSP 2011 - Proceedings*, 2011, pp. 19–23. doi: 10.1109/TSP.2011.6043785.

[27] Q. H. Dang, "Randomized hashing for digital signatures," Gaithersburg, MD, 2009. doi: 10.6028/NIST.SP.800-106.

[28] R. Sobti and G. Ganesan, "Cryptographic Primitives View project Advances in Web Crawler," 2012. [Online]. Available: https://www.researchgate.net/publication/267422045

[29] K. Talbott, "NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition," *NIST*, Oct. 02, 2012.

[30] J. W. Bos and D. Stefan, "Performance Analysis of the SHA-3 Candidates on Exotic Multi-Core Architectures," 2010. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/sha-3/index.html

[31] J.-P. Aumasson, S. Neves, Z. Wilcox-O'hearn, and C. Winnerlein, "LNCS 7954 - BLAKE2: Simpler, Smaller, Fast as MD5," 2013. [Online]. Available: https://blake2.net

[32] X. Wang, X. Lai, D. Feng, H. Chen, and X. Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD," 2005.

[33] Eli Biham and Rafi Chen, "Near-Collisions of SHA-0," 2004.

[34] F.-180-2 National Institute of Standards and Technology, "SECURE HASH STANDARD," 2002. [Online]. Available: http://csrc.nist.gov/publications/PubsFIPS.html#fips180-4.

[35] H. Yoshida and A. Biryukov, "LNCS 3897 - Analysis of a SHA-256 Variant," 2005.

[36] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The Keccak reference," 2011. [Online]. Available: http://keccak.noekeon.org/

[37] Jian Guo, MeiCheng Liu, and Ling Song, "Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak," vol. 10031, 2016, doi: 10.1007/978-3-662-53887-6.

[38] P. Emanuele, "Mining algorithms (Proof of Work): Blake2b, Equihash, Tensority and X16R & S," *https://en.cryptonomist.ch/*, Jul. 28, 2019.

[39] S. Atiwa, Y. Dawji, A. Refaey, and S. Magierowski, "Accelerated Hardware Implementation of BLAKE2 Cryptographic Hash for Blockchain," in *Canadian Conference on Electrical and Computer Engineering*, Institute of Electrical and Electronics Engineers Inc., Aug. 2020. doi: 10.1109/CCECE47787.2020.9255709.

[40] A. Hrytsak, V. Kinzeryavyy, D. Prysiazhnyi, Y. Burmak, and Y. Samoylik, "High-Speed And Secure Hash Function For Blockchain Security Mechanisms," *Scientific and Practical Cyber Security Journal (SPCSJ)*, vol. 4, no. 1, pp. 65–70, 2020.

[41] W. Jacob, "Hash Rate," *Investopedia* , Feb. 28, 2023.

[42] L. Klinger, L. Zhang, and Z. Zhou, "A Mean Field Game Analysis of Consensus Protocol Design," Aug. 2021, [Online]. Available: http://arxiv.org/abs/2108.09999

[43] Brian Nibley, "What Is a Good Hashrate?," *SoFi Learn*, Sep. 12, 2022.

[44] A. Kuznetsov, I. Oleshko, V. Tymchenko, K. Lisitsky, M. Rodinko, and A. Kolhatin, "Performance analysis of cryptographic hash functions suitable for use in blockchain," *International Journal of Computer Network and Information Security*, vol. 13, no. 2, pp. 1–15, Apr. 2021, doi: 10.5815/IJCNIS.2021.02.01.

[45] A. Maetouq *et al.*, "Comparison of Hash Function Algorithms Against Attacks: A Review," 2018. [Online]. Available: www.ijacsa.thesai.org

[46] F. Wang *et al.*, "An Experimental Investigation into the Hash Functions Used in Blockchains," *IEEE Trans Eng Manag*, vol. 67, no. 4, pp. 1404–1424, Nov. 2020, doi: 10.1109/TEM.2019.2932202.

[47] M. M. Özcan, B. A. Ayaz, M. M. Karagöz, And E. Yolaçan, "Performance Evaluation of SHA-256 and BLAKE2b in Proof of Work Architecture," *Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi*, May 2022, doi: 10.53608/estudambilisim.1086400.