EasyChair Preprint

№ 7689

Design of Hash Algorithm for Blockchain Security

Yatri Davda, R Sunitha and Prasad B Honnavalli

April 2, 2022

Design of Hash Algorithm for Blockchain Security

*Note: Sub-titles are not captured in Xplore and should not be used

Yatri Davda
dept. of Computer Science and Engg.
PES University
Bangalore, India
yndavda@gmail.com

Asst. Prof. Sunitha R.
dept. of Computer Science and Engg.
PES University
Bangalore, India
sunithar@pes.edu

Prof. Prasad B. Honnavalli
dept. of Computer Science and Engg.
PES University
Bangalore, India
prasadhb@pes.edu

***Abstract*—The BlockChain is that the invention that permits digitally generated information to be allocated without being copied. BlockChain Technology is that the heart of the new internet i.e., virtual currency. Emerging clever settlement structures over decentralized cryptocurrencies permit jointly suspicious events to transact competently without relied on third party, i.e., the reason to provide wide security to BlockChain Technology. Cryptography has so many algorithms to provide security such as MD5, AES, RSA, SHA family, etc. Hash functions are extremely useful and appear in almost all information security applications so, hashing techniques are more secure among them. We are designing a new approach i.e., SHA-512 in local blockchain application. SHA-512 is very secure algorithm uses 64-bit words and operates on 1024-bit blocks. We are proving that SHA-512 is more collision resistant than its predecessor with few mathematical models.**

***Index Terms*—blockchain, cryptocurrencies, SHA-512, reliability, timestamp**

I. Introduction

Today, some technologies are playing key role in the transformation of organizations like blockchain, IoT, AI and automation, to a cognitive enterprise. Interaction with its client's ecosystem which is used to find any enterprise as well as blockchain solutions to make these interactions efficient. Blockchains are additionally reliant on hashing. Hashing is a cryptographic strategy which is used to convert the data into a string of characters. Just as giving security through encryption, hashing makes a more proficient stockpiling of information, as the hash is of a fixed size. In view of providing better security, we are proposing a new idea based on the concept of Hashing in cryptography which aims at offering ownership protection in blockchain.

Blockchain refers to a technology that brings in the solution to the age-old human trust problem. It emerged in the market with the renowned cryptocurrency Bitcoin. It provides an architecture that allows us to trust on a decentralized system (Internet or Web) rather than trusting any actor within it. It runs on top of a peer to peer network and holds the identical copies of the ledger of transactions. This helps to avoid any middleman and the entire process of transaction takes place through machine consensus.

Identify applicable funding agency here. If none, delete this.

It is a ledger that is shared between multiple entities that everyone can inspect but not any single user can control it. It is a distributed cryptographically secured database that keeps the record of every transaction from the very initial one.

Blockchain mainly contains hash, hash of previous block and data, data can store an amount of money, a share in a company, a digital certificate of ownership, a vote during an election, or any other value.

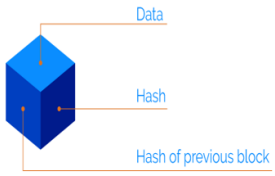

Fig. 1. basic of blockchain

The foundation of cryptographic money is the blockchain, which is a worldwide record framed by connecting individual squares of exchange information. The blockchain just contains approved exchanges, which forestalls deceitful exchanges and twofold expenditure of the money. The subsequent scrambled worth is a progression of numbers and letters that don't look like the first information and is known as a hash. Digital currency mining includes working with this hash.

Hashing requires processing the information from a block through a mathematical relation, which ends in an output of a set length. employing a fixed-length output increases security since anyone trying to decrypt the hash won't be ready to tell how long or short the input is just by watching the length of the output.

Solving the hash starts with the data available within the block header and is truly solving a posh mathematical problem. Each block header contains a version number, a timestamp, the hash utilized in the previous block, the hash of the Merkle Root, the nonce, and the target hash. The miner focuses on the nonce, a string of numbers. This number is appended to the hashed contents of the previous block, which is then hashed. If this new hash is a smaller amount than or up to the target hash, then it's accepted because the solution,

the miner is given the reward, and the block is added to the blockchain.

The approval cycle for blockchain exchanges depends on information being encoded utilizing algorithmic hashing. Comprehending the hash requires the digger to figure out which string to use as the nonce, which itself requires a lot of experimentation.

It is profoundly impossible that an excavator will effectively think of the right nonce on the main take a stab at, implying that the digger may possibly test numerous nonce choices prior to taking care of business. The more prominent the trouble—a proportion of the fact that it is so difficult to make a hash that meets the prerequisite of the objective hash—the more it is probably going to take to create an answer.

II. Ease of Use

A. Novelty

SHA-2 hash algorithms are distinguished by the length of the output they produce. The two basic variants are SHA-256 and SHA-512, which are in fact the same algorithm, applied to different word lengths. SHA-256 operates on 32-bit words, whereas SHA-512 works on 64-bit words. The two variants differ also in some constant parameters and values, and employ different initialization values. The other four versions, namely SHA-224, SHA-384, SHA-512/224, and SHA-512/256, are the same function as SHA-256 (the first one) or SHA-512 (the others), with the output truncated to the specified number of bits and different initial values. The whole family can hence be described by looking only at the two basic variants.

B. Security

SHA-512/256 which simply truncates the SHA-512 output to the same size as SHA-256's output. This allows the hash to be more efficiently implemented, but not require more space to store. Although this variant came later, probably the initial design of SHA-512 was to take advantage of 64-bit processors. A larger hash also serves as a more secure variant; although no attacks against full-round SHA-2 are known, the extra 256 bits gives quite a lot of headroom to extend the life of SHA-512 even if effective attacks are made against SHA-256. SHA-512 is more collision resistant than it's predecessor SHA-256 and SHA-128 based on mathematical model.

C. Reliability

The reliability and integrity of blockchain is rooted in there being no chance of any fraudulent data or transactions, such as a double spend, being accepted or recorded. A cornerstone of the technology as a whole and the key components in maintaining this reliability is hashing. This is one to one conversation i.e., not depends on any third-party. For e.g., in the case of money transaction any causes happens then it will be depends on particular bank but in the case of cryptocurrencies it is one way communication without involving third-party.

D. Application compatibility

The algorithm is going to be compatible with applications and filesystems. It may be compatible with databases, any OS (Windows/Linux). The algorithm in blockchain technology is making waves in several industries, including:

- Finance
- Music and entertainment
- Diamond and precious assets
- Artwork
- Supply chains of various commodities

E. Reusability

One of the essential difficulties that any new innovation faces are ease of use. This issue is more acute in blockchain because of new architecture and high stakes. The transaction flow should be visible to users to analyze the whole transaction flows. This will improve the usability and help the individuals to understand and analyze the whole blockchain network.

Literature Survey

Some of the popular algorithms are offering secure transaction with fixed length are SHA-256. This is always the case whether the transaction is just a single word or a complex transaction with huge amounts of data.

Hashing in blockchain refers to the process of having an input item of whatever length reflecting an output item of a fixed length. If we take the example of blockchain use in cryptocurrencies, transactions of varying lengths are run through a given hashing algorithm, and all give an output that is of a fixed length. This is regardless of the length of the input transaction.

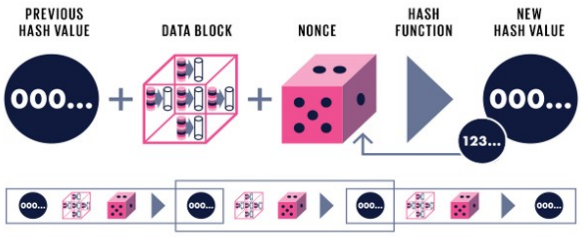

Fig. 2. Basic hashing process

The size of the hash will depend on the hash function utilized, but the out using a particular hashing algorithm will be of a specific size. For e.g., when we will consider one YouTube video say 50mb and we are applying SHA-256 algorithm on it then the output will be hash of 256bits in length which is really a consistent hashing.

Hashing algorithm must fulfill specific criteria to be an effective cryptographic algorithm. The same input must always generate the same output. Regardless of how many times you put the data through the hashing algorithm, it must consistently produce the same hash with identical characters in the string.

The input cannot be deduced or calculated using the output. There should be no way to reverse the hashing process to see

the original data set. Any change in the input must produce an entirely different output. Even changing the case of one character in a data set should create a hash that is significantly different. The hash should be of a fixed number of characters, regardless the size or type of data used as an input. Creating the hash should be a fast process that doesn't make heavy use of computing power.

Cryptography is a process of protecting information using codes so that, only those for whom the information is, can read and process it. Hashing drastically increases the security of the data. There is no way to decrypting the data because we are not encrypting it. As I mentioned already it's a one-way cryptographic function. A cryptographic hash function needs to have several crucial qualities to be considered useful, these include:

- Every hash is different from another.
- Same Hash value will be always produced for the same message.
- Impossible to decide input based on the hash value.
- Even a small change to the input whole hash will be changed.

Hashing helps us to see if the data has tampered or not. Hashing is of the core fundamentals and foremost aspects of the immutable and defining potential of blockchain technology.

A new hashing algorithm based on logistics aims at providing computationally efficient and effective algorithm for securing bitcoin transactions in blockchain. It is based on the concept of designing new hash algorithm using the blockchain concept. To perform the hashing in blockchain We get our answer on the first try. The odds of this happening are astronomical.

The working principle of this algorithm is as follows,

- A hash is a function that meets the encrypted demands needed to solve for a blockchain computation.
- A hash, like a nonce or a solution, is the backbone of the blockchain network.
- Hashes are of a fixed length since it makes it nearly impossible to guess the length of the hash if someone was trying to crack the blockchain.
- A hash is developed based on the information present in the block header.

Although, this idea was proposed to secure transaction during the transformation in organization. The validation process for blockchain transactions relies on data being encrypted using algorithmic hashing.

Limitations of Existing Model

The fact that the output of a hash function cannot be reverted back to the input using an efficient algorithm does not mean that it cannot be cracked. Databases containing hashes of common words and short strings are usually within our reach with a simple google search. Also, common strings can be easily and quickly brute-forced or cracked with a dictionary attack.

51% attack: In the 51% attack, if an entity can control 51% or more of the network nodes, then it can result in control of the network. By doing so, they can modify the data in the ledger and also do double-spending. This is possible on networks where the control of miners or nodes are possible. This means that private networks are more likely to be safe from 51% attacks, whereas public ones are more vulnerable to this.

Double-Spending: Double-spending is yet another problem with the current blockchain technology. To prevent double-spending the blockchain network deploys different consensus algorithms including Proof-of-Stake, Proof-of-Work, and so on. Double spending is only possible on networks with a vulnerability to the 51

DDoS's Attack: In a DDoS attack, the nodes are bombarded with similar requests, congesting the network and bringing it down.

Cryptographic Cracking: Another way the blockchain technology is not secure is that the cryptographic solution that it utilizes. Quantum algorithms or computing are more than capable of breaking cryptographic cracking. However, blockchain solutions are now implementing quantum-proof cryptographic algorithms.

Users Are Their Own Bank: Private Keys: To make blockchain decentralized, it is important to give individuals the ability to act as their own bank. However, this also leads to another problem. To access the assets or the information stored by the user in the blockchain, they need private keys. It is generated during the wallet creation process, and it is the responsibility of the user to take proper note of it. They also need to make sure that they do not share it with anyone else. If they fail to do so, their wallet is in danger. Also, if they lose the private key, they will lose access to the wallet forever. The reliance on users makes it as one of the disadvantages of blockchain.

Interoperability: There are multiple types of blockchain networks which work differently, trying to solve the DLT problem in their own unique way. This leads to interoperability issues where these chains are not able to communicate effectively. The interoperability issue also persists when it comes to traditional systems and systems using blockchain technology.

Legacy Systems: Not all businesses have changed from legacy systems. There are still many organizations that rely on legacy systems to run their business. However, if they want to adopt blockchain technology, they need to completely get rid of their systems and change to blockchain technology — which is not feasible for every business out there.

Proposed Methodology

Hashing requires processing the data from a block through a mathematical function, which results in an output of a fixed length. Using a fixed-length output increases security since anyone trying to decrypt the hash won't be able to tell how long or short the input is simply by looking at the length of the output.

Solving the hash starts with the data available in the block header and is essentially solving a complex mathematical problem. Each block header contains a version number, a

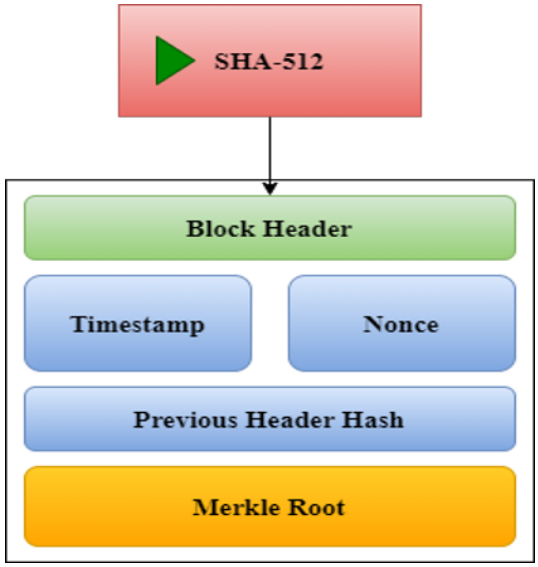

Fig. 3. Block Header

timestamp, the hash used in the previous block, the hash of the Merkle Root, the nonce, and the target hash.

Understanding a Merkle Root A blockchain is comprised of various blocks that are linked with one another (hence the name blockchain). A hash tree, or the Merkle tree, encodes the blockchain data in an efficient and secure manner. It enables the quick verification of blockchain data, as well as quick movement of large amounts of data from one computer node to the other on the peer-to-peer blockchain network.

Every transaction occurring on the blockchain network has a hash associated with it. However, these hashes are not stored in a sequential order on the block, rather in the form of a tree-like structure such that each hash is linked to its parent following a parent-child tree-like relation.

Since there are numerous transactions stored on a particular block, all the transaction hashes in the block are also hashed, which results in a Merkle root.

For example, consider a seven-transaction block. At the lowest level (called the leaf-level), there will be four transaction hashes. At the level one above the leaf-level, there will be two transaction hashes, each of which will connect to two hashes that are below them at the leaf level. At the top (level two), there will be the last transaction hash called the root, and it will connect to the two hashes below it (at level one).

Effectively, you get an upside-down binary tree, with each node of the tree connecting to only two nodes below it (hence the name "binary tree"). It has one root hash at the top, which connects to two hashes at level one, each of which again connects to the two hashes at level three (leaf-level), and the structure continues depending upon the number of transaction hashes.

The hashing starts at the lowest level (leaf-level) nodes, and all four hashes are included in the hash of nodes that are linked to it at level one. Similarly, hashing continues at level one, which leads to hashes of hashes reaching to higher levels, until it reaches the single top root hash.

This root hash is called the Merkle root, and due to the tree-like linkage of hashes, it contains all the information about every single transaction hash that exists on the block. It offers a single-point hash value that enables validating everything present on that block.

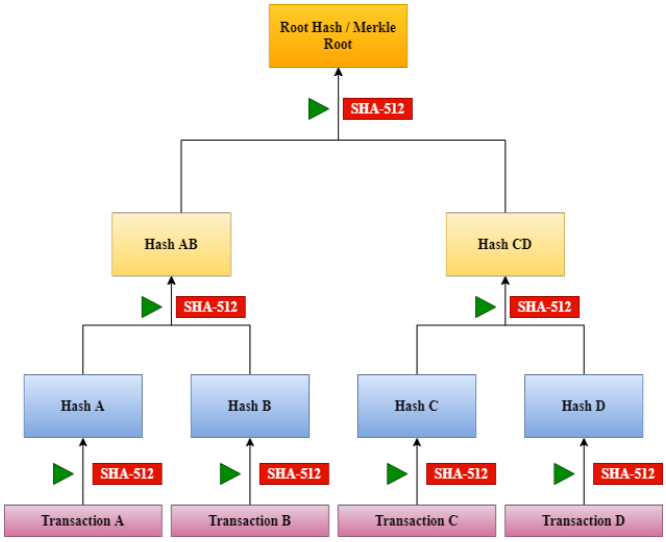

Fig. 4. Block diagram of Merkle Root with SHA-512

The miner focuses on the nonce, a string of numbers. This number is appended to the hashed contents of the previous block, which is then hashed. If this new hash is less than or equal to the target hash, then it is accepted as the solution, the miner is given the reward, and the block is added to the blockchain.

Solving the hash requires the miner to determine which string to use as the nonce, which itself requires a significant amount of trial-and-error. This is because the nonce is a random string. It is highly unlikely that a miner will successfully come up with the correct nonce on the first try, meaning that the miner may potentially test a large number of nonce options before getting it right. The greater the difficulty—a measure of how hard it is to create a hash that meets the requirement of the target hash—the longer it is likely to take to generate a solution.

To proving that SHA-512 is more collision resistant then its predecessor, let's see the concept of pigeonhole principle.

Pigeonhole Principle: The pigeonhole principle states that, if k is a positive integer & $k+1$ or more objects are placed into k boxes, then there is at least one box containing two or more of the objects.

Assume,

$$M = 6\ bits,\ D = 4\ bits$$

$$\therefore\ Possible\ no.\ of\ digests(pigeonholes)\ is\ 2^4 = 16$$

$$\therefore\ Possible\ no.\ of\ message(pigeons)\ is\ 2^6 = 16$$

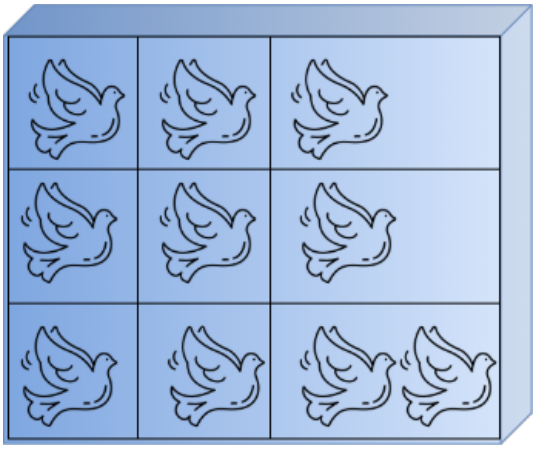

Fig. 5. pigeonhole principle

Let's consider possible no. of digests as n & Possible no. of message as m.

$$So,\ n = 16$$
$$\therefore\ kn + 1 = 64$$
$$\therefore\ 4(16) + 1 = 64$$
$$So,\ k > 3,\ \ \therefore\ 4 > 3$$
$$\therefore\ 4(k+1)\ message$$

Now, based on the above example we will find the probability of collision in SHA-160, SHA-256 & SHA-512.

SHA-160:
Let's take

$$M = 2048\ bits,\ in\ SHA-160\ \ D = 160\ bits$$
$$\therefore\ Possible\ no.\ of\ digests(pigeonholes)\ is\ 2^{160}$$
$$\therefore\ Possible\ no.\ of\ message(pigeons)\ is\ 2^{2048}$$
$$So,\ n = 2^{160}$$
$$\therefore\ kn + 1 = 2^{2048}$$
$$\implies\ k(2^{160}) + 1 = 2^{2048}$$
$$\implies\ (2^{1888}\ \times\ 2^{160}) + 1 = 2^{2048}$$
$$\implies\ \mathbf{2^{1888}}(k+1)\ message$$

SHA-256:
Let's take

$$M = 2048\ bits,\ in\ SHA-256\ \ D = 256\ bits$$
$$\therefore\ Possible\ no.\ of\ digests(pigeonholes)\ is\ 2^{256}$$
$$\therefore\ Possible\ no.\ of\ message(pigeons)\ is\ 2^{2048}$$
$$So,\ n = 2^{256}$$
$$\therefore\ kn + 1 = 2^{2048}$$
$$\implies\ k(2^{256}) + 1 = 2^{2048}$$
$$\implies\ (2^{1792}\ \times\ 2^{256}) + 1 = 2^{2048}$$
$$\implies\ \mathbf{2^{1792}}(k+1)\ message$$

SHA-512:
Let's take

$$M = 2048\ bits,\ in\ SHA-512\ \ D = 512\ bits$$
$$\therefore\ Possible\ no.\ of\ digests(pigeonholes)\ is\ 2^{512}$$
$$\therefore\ Possible\ no.\ of\ message(pigeons)\ is\ 2^{2048}$$
$$So,\ n = 2^{512}$$
$$\therefore\ kn + 1 = 2^{2048}$$
$$\implies\ k(2^{512}) + 1 = 2^{2048}$$
$$\implies\ (2^{1536}\ \times\ 2^{512}) + 1 = 2^{2048}$$
$$\implies\ \mathbf{2^{1536}}(k+1)\ message$$

Here, we can say that SHA-512 is more collision resistant than SHA-160 & SHA-256 because probability of collisions in SHA-512 is less than the probability of collisions in SHA-160 & SHA-256. (since $2^{1888} > 2^{1792} > \mathbf{2^{1536}}$).

IMPLEMENTATION

Fundamentally, the data stored in a blockchain must have the following characteristics:

- Immutable
- Unhackable
- Persistent (no loss of data)
- Distributed

I will assume that the data stored in the block is transactional data.

```
Step 1: Create a single block
    Block:
        init (
            self, index,
            transactions,
            timestamp,
            previous_hash, nonce)
        {
            self.index = block index
                self.transactions =
                    Any data (Assume,
                        transnational
                            data)
                self.timestamp =
                    block creation time
                If self.previous_hash
                    is not NULL
            Then, it is hash value
                of the previous block
        }

Step 2: SHA-512 can be implemented by
    adding a compute_hash method.
        compute_hash (self)
```

Now, that we've established a single block, we need a way to chain them together

```
Step 3: We also need a way to initialize
the blockchain, so we define the create_
genesis_block method. This creates an
initial block with an index of 0 and a
previous hash of 0. We then add this to
the list chain that keeps track of each
block.

    Blockchain:
        create_genesis_block ()
            {
                genesis_block =
                    Block(0, [],
                        time.time(), "0")
                genesis_block.hash =
                genesis_block.
                            compute_hash()
                self.chain.append
                    (genesis_block)
            }

    last_block(self):
        return self.chain[-1]

Step 4: Proof-of-work system for
blockchain
        difficulty = 2
        Proof_of_work (block)
        {
                Computed_hash =
                    block.compute_hash()
                While not
                    computed_hash.
                    startswith (0 *
                    Blockchain.difficulty)
                {
                        Computed_hash =
                            block.
                            compute_hash()
                }
                Return computed_hash
        }

Step 5: Add block to the chain
        Check proof is valid or not
        Add new transaction if any

Step 6: Mining process
        Mine the new block i.e.,
            add_block()
```

REFERENCES

[1] S. Dhumwad, M. Sukhadeve, C. Naik, M. K.N. and S. Prabhu, "A Peer to Peer Money Transfer Using SHA256 and Merkle Tree," 2017 23RD Annual International Conference in Advanced Computing and Communications (ADCOM), 2017, pp. 40-43, doi: 10.1109/ADCOM.2017.00013.

[2] M. Nehe and S. A. Jain, "A Survey on Data Security using Blockchain: Merits, Demerits and Applications," 2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC), 2019, pp. 1-5, doi: 10.1109/ICRAECC43874.2019.8995064.

[3] L. Bauer, M. Shafique and J. Henkel, "RISPP: A run-time adaptive reconfigurable embedded processor," 2009 International Conference on Field Programmable Logic and Applications, 2009, pp. 725-726, doi: 10.1109/FPL.2009.5272323.

[4] Ishan, P. Bhulania and G. Raj, "Analysis of Cryptographic Hash in Blockchain for Bitcoin Mining Process," 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE), 2018, pp. 105-110, doi: 10.1109/ICACCE.2018.8441688.

[5] S. Singh and N. Singh, "Blockchain: Future of financial and cyber security," 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), 2016, pp. 463-467, doi: 10.1109/IC3I.2016.7918009.

[6] R. Aswini and K. Kiruba, "College Fees Transaction Using Hash Functions of Blockchain Model," 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), 2019, pp. 1-6, doi: 10.1109/ICSCAN.2019.8878769.

[7] A. A. Alkandari, I. F. Al-Shaikhli and M. A. Alahmad, "Cryptographic Hash Function: A High Level View," 2013 International Conference on Informatics and Creative Multimedia, 2013, pp. 128-134, doi: 10.1109/ICICM.2013.29.

[8] N. Lee, J. Yang, M. M. H. Onik and C. Kim, "Modifiable Public Blockchains Using Truncated Hashing and Sidechains," in IEEE Access, vol. 7, pp. 173571-173582, 2019, doi: 10.1109/ACCESS.2019.2956628.

[9] D. K.N. and R. Bhakthavatchalu, "Parameterizable FPGA Implementation of SHA-256 using Blockchain Concept," 2019 International Conference on Communication and Signal Processing (ICCSP), 2019, pp. 0370-0374, doi: 10.1109/ICCSP.2019.8698069.

[10] J. Ferreira, M. Antunes, M. Zhygulskyy and L. Frazão, "Performance of Hash Functions in Blockchain Applied to IoT Devices," 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), 2019, pp. 1-7, doi: 10.23919/CISTI.2019.8760885.

[11] A. L. Selvakumar and C. S. Ganadhas, "The Evaluation Report of SHA-256 Crypt Analysis Hash Function," 2009 International Conference on Communication Software and Networks, 2009, pp. 588-592, doi: 10.1109/ICCSN.2009.50.

[12] R. N. A. Sosu, K. Quist-Aphetsi and L. Nana, "A Decentralized Cryptographic Blockchain Approach for Health Information System," 2019 International Conference on Computing, Computational Modelling and Applications (ICCMA), 2019, pp. 120-1204, doi: 10.1109/ICCMA.2019.00027.

[13] W. Ao, S. Fu, C. Zhang, Y. Huang and F. Xia, "A Secure Identity Authentication Scheme Based on Blockchain and Identity-based Cryptography," 2019 IEEE 2nd International Conference on Computer and Communication Engineering Technology (CCET), 2019, pp. 90-95, doi: 10.1109/CCET48361.2019.8989361.

[14] E. E. Ramos and D. Pour Yousefian Barfeh, "Advanced Encryption Standard-Cipher Blockchain Mode for File Cryptography Stint Control," 2019 5th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), 2019, pp. 1-5, doi: 10.1109/ICSPIS48872.2019.9066142.

[15] L. Xin and B. Zhang, "Application of Blockchain News Production Based on Digital Encryption Technology," 2020 International Conference on Computer Engineering and Application (ICCEA), 2020, pp. 853-856, doi: 10.1109/ICCEA50009.2020.00187.

[16] V. Hassija, M. Zaid, G. Singh, A. Srivastava and V. Saxena, "Cryptober: A Blockchain-based Secure and Cost-Optimal Car Rental Platform," 2019 Twelfth International Conference on Contemporary Computing (IC3), 2019, pp. 1-6, doi: 10.1109/IC3.2019.8844943.

[17] H. Nieto-Chaupis, "Description of Processes of Blockchain and Cryptocurrency with Quantum Mechanics Theory," 2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON), 2019, pp. 1-4, doi: 10.1109/CHILECON47746.2019.8988006.

[18] G. George and S. Sankaranarayanan, "Light weight Cryptographic solutions for Fog Based Blockchain," 2019 International Conference on Smart Structures and Systems (ICSSS), 2019, pp. 1-5, doi: 10.1109/ICSSS.2019.8882870.

[19] M. Sato and S. Matsuo, "Long-Term Public Blockchain: Resilience against Compromise of Underlying Cryptography," 2017 26th International Conference on Computer Communication and Networks (ICCCN), 2017, pp. 1-8, doi: 10.1109/ICCCN.2017.8038516.
[20] T. M. Fernández-Caramès and P. Fraga-Lamas, "Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks," in IEEE Access, vol. 8, pp. 21091-21116, 2020, doi: 10.1109/ACCESS.2020.2968985.
[21] "Wallets and Crypto Terminals for Blockchain; Introducing the Big Bang paradigm," 2019 Fifth Conference on Mobile and Secure Services (MobiSecServ), 2019, pp. 1-1, doi: 10.1109/MOBISECSERV.2019.8686561.
[22] S. Homayoun, A. Dehghantanha, R. M. Parizi and K. -K. R. Choo, "A Blockchain-based Framework for Detecting Malicious Mobile Applications in App Stores," 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), 2019, pp. 1-4, doi: 10.1109/CCECE.2019.8861782.
[23] T. P. Keenan, "Alice in Blockchains: Surprising Security Pitfalls in PoW and PoS Blockchain Systems," 2017 15th Annual Conference on Privacy, Security and Trust (PST), 2017, pp. 400-4002, doi: 10.1109/PST.2017.00057.
[24] P. Pal and S. Ruj, "BlockV: A Blockchain Enabled Peer-Peer Ride Sharing Service," 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 463-468, doi: 10.1109/Blockchain.2019.00070.
[25] X. Yang, Y. Chen and X. Chen, "Effective Scheme against 51% Attack on Proof-of-Work Blockchain with History Weighted Information," 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 261-265, doi: 10.1109/Blockchain.2019.00041.
[26] S. Linoy, H. Mahdikhani, S. Ray, R. Lu, N. Stakhanova and A. Ghorbani, "Scalable Privacy-Preserving Query Processing over Ethereum Blockchain," 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 398-404, doi: 10.1109/Blockchain.2019.00061.