The originality of this thesis has been checked in accordance with the University of Turku quality assurance system using the Turnitin Originality Check service.

# Overview of Blockchain Technology Cryptographic Security

Nwachukwu, Uchechukwu Justin

Master of Science Thesis
March 2019

Department of Mathematics and Statistics
University of Turku

UNIVERSITY OF TURKU

Department of Mathematics and Statistics

NWACHUKWU, Uchechukwu Justin:

Overview of Blockchain Technology Cryptographic Security

Master of Science Thesis, 47 pages,

Information Security and Cryptography

March 2019

---

This thesis work is aimed at developing understanding of the hash functions and algorithms being used in blockchain technologies Bitcoin in comparison to Ethereum and private blockchain hash functions. This study attempts to answer one fundamental research question: "What considerations are important in assessing blockchain cryptographic security, with an emphasis on hash functions".

The study was carried out qualitatively using a desk research approach and combining this approach with using two public blockchains-based cryptocurrencies; Ethereum and Bitcoin as case studies. The research aims to provide a holistic view of blockchain cryptographic security comparing Bitcoin and Ethereum as use cases, and thus providing a consolidated document which students studying cryptography can access to obtain a better understanding of what is involved in blockchain security. From an academic perspective, the research aims at providing a model which can be used in assessing what is important to consider in the cryptographic security of blockchains.

Three main categories of factors considered were presented in the proposed model which were strategical factors, complexity attributes and technical drivers. This results in a base crucial metrics such as absence of secret seeds, efficiency of verification, preimage collision resistance, fixed output size, low collision probability, and even distribution of preimages in output.

*Keywords:* Blockchain, Hash Functions, Digital Signatures, Consensus Algorithms, Bitcoin, Ethereum, Hashing Security Factors

# Table of Contents

# LIST OF FIGURES

# 1. INTRODUCTION

## 1.1 Motivation and Background

“A violent ocean filled with gold plated fishes” - this effectively describes the current technological trend in the form of blockchain and crypto-currencies, which has enormous profit potential, but also has so many uncertainties and potential risks which can prove fatal economically if they are not considered. In 2014 alone, according to the leaked Mt. Gox "crisis strategy draft" document, a Bitcoin exchange, Mt. Gox, filed for bankruptcy after losing more than 700,000 Bitcoins worth over 400 million USD (Kaushal *et al*., 2017). In general, the applications of blockchain technology and their associated value has increased threefold since 2016, and the sustainability of such steep and rapid increments concerns sceptical observers and users. This creates a need to understand the core and founding principles these technologies are built upon in order to be able to design efficient risk management strategies and processes. Having such strategies in place would help mitigate losses and circumvent huge waves of financial crisis for cryptocurrency miners and users.

The case studies of this research, namely Ethereum and Bitcoin cryptocurrencies are built upon the foundation of blockchain technologies which are distributed peer to peer public ledger systems and which according to De Meijer (2015) has the potential of changing the financial world, with a wide and diverse range of implementations in digital payments and smart contracts. Hence, the research on cryptocurrencies have an associated and comparable impact on blockchain technologies as well.

## 1.2 Research contribution

As the name implies, a blockchain in very simple terms is a chain of blocks which have data stored in them, and essentially a distributed ledger or record of transactions with each block in the chain cryptographically linked to its predecessor as in Figure 1.


**Figure 1:** A Blockchain with “n+1” blocks

The concepts discussed in this thesis are topics which have been around for quite some time and thus there has been ongoing discussions in previous literary works concerning these concepts such

as blockchain, Bitcoin, digital signatures, hash functions and time-stamping techniques dating back as far as 1991 in Haber's work "How to time-stamp a digital document" (Haber and Stornetta, 1991). However, research seeking to fuse these concepts together remains scarce.

This study attempts to answer one fundamental research question which is; what considerations are important in assessing blockchain cryptographic security particularly hash functions? The study uses public blockchain systems; Ethereum and Bitcoin as case studies.

For the purpose of this research, Bitcoin and Ethereum cryptocurrencies are profoundly studied with a qualitative desk research approach to a case study methodology due to the diverse nature and relatively young age of the topic area. Ethereum and Bitcoin were selected as case studies as these represented two major differing implementations regarding consensus and decision making while maintaining some similarity in certain areas such as digital signing. This enabled obtaining an in-depth understanding of security of hash functions which Bitcoin and Ethereum uses as well as justification behind their choices, and its interactions with other aspects of the technology.

This research aims to provide a holistic view of blockchain cryptographic security, and thus providing a consolidated document which students studying cryptography can access to obtain better understanding of what is involved in blockchain security and hence be able to have a positive impact in the design and implementations of blockchain technology systems and their corresponding risk mitigation processes. From an academic perspective, the research aims to provide a model which can be used in assessing what is important to consider in the security of blockchain technology.

### 1.3 Research organization

This literary work is organized into six main sections. In the first section, the thesis was introduced with a summary of the research motivations and the research's contributions to the academic and practical audience. In chapter two, the basic mathematical notations utilized in this study, as well as an overview of the mathematical structures and functions which form the core of how blockchain cryptography were introduced. These provided a mathematical basis to build upon in later chapter, particularly in chapter five. Similarly, a brief review of the previous literary works pertaining blockchain technology was provided in later sections of chapter two and served as an

introduction to the key blockchain sub-topics. Chapter three provided an explanation to the methodological choices of this research.

In chapter four of this literary work, as a prelude to blockchain cryptography, a structural analysis of blockchains was presented. Subsequently, in chapter five, the main concepts which form the core of blockchain security and privacy were introduced with detailed explanation of hashing and digital signature algorithms. Cryptocurrencies; Bitcoin and Ethereum, were explored to a greater extent in chapter six, and their cryptographic security, internal structure and innerworkings were discussed. The intended goal of chapter six was to provide a deeper understanding of the chosen cryptocurrencies and reasoning behind the cryptographic choices in an attempt to extract some considerations which could be generalized for other blockchain-based systems. This culminated in one of the main goals of this research which is the creation of an assessment model for cryptographic hash functions.

# 2. THEORETICAL BACKGROUND

## 2.1 Mathematical notations and preliminary concepts

The core mathematical concepts discussed in this research requires some knowledge of basic mathematics number theory principles and provides an overview of the mathematical structures and functions which form the core of blockchain security and privacy.

Modular arithmetic is one of the basic concepts that is central to many mathematical calculations in blockchain technologies, such as finite fields and consequently elliptic curves which are defined over finite fields. This is used often when describing hashes and digital signatures in later chapters.

**Definition 2.1.** Let "**a**", "**b**", and "**m**" be numbers from the set of all integers; "**a**" and "**b**" are congruent to the modulus "**m**" if they both have the same value "**r**" as a remainder when they are divided by "**m**" (Paar and Pelzl, 2010).

$$\forall\, \boldsymbol{x, r, m} \in \boldsymbol{Z}\ where\ \boldsymbol{m} > 0;\ \boldsymbol{x} \equiv \boldsymbol{r}\ (mod\ \boldsymbol{m})\ if\ \boldsymbol{m} \mid (\boldsymbol{x} - \boldsymbol{r})$$

For example, in modulo 7, 26 would be congruent to 5; that is, **26 ≡ 5** (mod *7*), meaning if "26 – 5" which is 21 can be divided by 7.

**Theorem 2.1.** Basically, it is possible to write any integer x ∈ ℤ in the form;

$$\boldsymbol{x = q \times m + r\ where\ 0 \le r < m} \quad (\boldsymbol{that\ is, r} \in \{\boldsymbol{0, 1, 2, 3, \dots, m-1}\})$$

The number 31 for example hence can be written as 31 = 4×7 + 3; and therefore 31 ≡ 3 (mod **7**). Although in theory, for every given modulus there are infinitely many possible values for the remainder ***r***, the same number 31 is congruent to 3 or 10 or 17 or -4 or -11 or other possibilities.

Some other basic concepts which will be introduced here include groups, fields and elliptic curves.

**Definition 2.2.** A group **G** is a set of elements which are closed under an operation " ∘ ", such that

$$\boldsymbol{x} \circ \boldsymbol{y} = \boldsymbol{z} \in \boldsymbol{G};\ \forall\, \boldsymbol{x, y} \in \boldsymbol{G}$$

$$(\boldsymbol{x} \circ \boldsymbol{y}) \circ \boldsymbol{z} = \boldsymbol{x}\,(\boldsymbol{y} \circ \boldsymbol{z});\ \forall\, \boldsymbol{x, y, z} \in \boldsymbol{G}\ (\text{associativity})$$

$$\exists \boldsymbol{e}\colon (\boldsymbol{x} \circ \boldsymbol{e}) = (\boldsymbol{x} \circ \boldsymbol{e}) = \boldsymbol{x};\ \forall\, \boldsymbol{x, y} \in \boldsymbol{G}$$

$$\exists\, \boldsymbol{x^{-1}}\colon (\boldsymbol{x} \circ \boldsymbol{x^{-1}}) = (\boldsymbol{x} \circ \boldsymbol{x^{-1}}) = \boldsymbol{e}; \forall \boldsymbol{x} \in \boldsymbol{G}$$

If **G** is also commutative such that **(x ∘ y) = (y ∘ x); ∀ x, y ∈ G,** then **G** is an abelian group.

It is useful to note that the order of an element in a group is the least integer "**k**" which the given element has to multiplied "**k**" times to obtain the identity element; $x \bullet x \bullet x \bullet \ldots \bullet x \bullet x = x^{k} = e$**.** A cyclic group is one which has an element with the maximum order, and such element is called a primitive element or generator and can generate every element in the group (Paar and Pelzl, 2010).

**Definition 2.3.** A **Field** is a set **F,** which satisfy the following properties:

$$(\boldsymbol{F}, +) \text{ is a commutative group with an identity element } \mathbf{0}$$

$$(\boldsymbol{F} \setminus \{0\}, \bullet) \text{ is a commutative group with an identity element } \mathbf{1}$$

$$(\boldsymbol{x} + \boldsymbol{y}) \bullet \boldsymbol{z} = (\boldsymbol{x} \bullet \boldsymbol{z}) + (\boldsymbol{y} \bullet \boldsymbol{z}) \; \forall \, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \boldsymbol{F} \text{ (distributivity)}$$

A field is said to be a finite field or Galois field if it contains a finite number of elements (Menezes *et al*., 1996) and the number of elements in the field is called order of the field.

The last concept to be introduced in this section is the elliptic curve (Paar and Pelzl, 2010).

**Definition 2.4** An elliptic curve **E** for cryptographic applications is the set of solutions to the following equation:

$$\boldsymbol{y^2 = x^3 + ax + b} \qquad \boldsymbol{/F_p}$$
$$\boldsymbol{where\ p > 3\ and\ a, b \in F_p :\ 4a^3 + 27b^2 \neq 0\ (mod\ p)}$$

In order words, it is the set of pairs or points $(\boldsymbol{x}, \mathbf{y}) \in \mathbf{Z_p}$ which satisfy the given equation above ($y^2 = x^3 + ax + b$; over a prime finite field $\boldsymbol{F_p}$). Adding of points in an elliptic curve, takes two different forms for when the points being added are equal and when the points are different. If point P = $(\boldsymbol{x_1}, \mathbf{y_1})$ and Q = $(\boldsymbol{x_2}, \mathbf{y_2})$, then $(\boldsymbol{x_1}, \mathbf{y_1}) + (\boldsymbol{x_2}, \mathbf{y_2})$ gives a new point R = $(\boldsymbol{x_3}, \mathbf{y_3})$, such that;

$$\boldsymbol{x_3 = s^2 - x_1 - x_2 \ \ and \ \ y_3 = s\,(x_1 - x_3) - y_1}$$

$$\boldsymbol{where\ \ s = \frac{(y_2 - y_1)}{(x_2 - x_1)}\ mod\ p;\ if\ P \neq Q}$$

$$\boldsymbol{and\ \ s = \frac{(3x_1^2 + a)}{2y_1}\ \ mod\ p;\ if\ P = Q}$$

## 2.2 Blockchain

This section aims to provide a light overview and introduction to the topics discussed at length in subsequent chapters, and to provide a glimpse into what has been said in the past and ongoing discussions by various authors on the key concepts discussed in this thesis. These concepts will be discussed at length from a more technical and security perspective in later chapters dedicated to each main topic grouping.

Blockchain was first conceptualized literally by Haber in their work on digital time stamping (Haber and Stornetta, 1991), before being put into implemented by Shatoshi in his white paper on Bitcoin cryptocurrency (Nakamoto, 2009) and subsequently popularized. Woodside *et al.* (2017) defined blockchain a digitized decentralized peer to peer transaction ledger and attempts to explain that blockchain has various underlying technologies and applications. Interestingly, Hanifatunnisa and Rahardjo (2017) described blockchain technologies as trustworthy service system to some non-trusting nodes/parties which ushers in the importance of security in blockchain technologies.

## 2.3 Blockchain types

According to Hanifatunnisa and Rahardjo (2017), blockchain technologies could be easily grouped into three main groups based on the way in which permissions are granted, and these main types are permissionless blockchain, public permissioned blockchain, and private permissioned blockchain.

### 2.3.1 Permissioned blockchain

The public version of permissioned blockchains have known and pre-approved participants/nodes and has the capability of identifying which participants are able to jointly modify or update the data in blocks and/or can control the data, and similarly there is a restriction on who can carry out transactions (Hanifatunnisa and Rahardjo, 2017). These sometimes are called federated or consortium blockchains. Private permissioned blockchains exclusively define who can participate in, access or view blockchain data and participation is strictly by invitation or permission, while in contrast public permissioned blockchain are at least accessible to the public for viewing. In general, the permissions control for a blockchain are defined by the security policy which they implement.

Based on the description of permissioned blockchain provided by Sato and Himura (2018), it could be suggested that permissioned blockchains are a better fit for consortia, enabling the opportunity of cross-organizational sharing of information and participation in business transactions. This is reinforced by the description of smart contracts as user defined business logics used in executing business transactions and which are based on a distributed consensus algorithm or as Letourneau and Whelan (2017) describes computer codes which make simple transactions automatically take place.

One good example of a permissioned blockchain is the Hyperledger Fabric. In their literary work, Pustišeka and Kos (2017) explained how the Hyperledger project came about as an attempt to have a framework of an "open source distributed ledger" which enables customizing via coding and allows using or plugging-into different or customized consensus algorithms. This makes it feasible to use the Hyperledger fabric as a private blockchain also in businesses.

### 2.3.2 Permissionless blockchain

In permissionless blockchain, participating in a consensus is generally open to anyone who is interested. Transaction blocks are generated by participants (miners) in what is called a mining process, and they validate the blocks and attach them to the chain using a consensus protocol (Dinh *et al.* 2017), and this creates security and performance challenges due to the byzantine nature of such a data structure, in the sense that there is an unreliable or incomplete knowledge as to if one of the nodes in the data structure has failed.

In Bitcoin there is an attempt to avoid this by indicating confirmation of a block only when its preceded by six confirmed blocks though this creates a security issue of its own where a powerful adversary can thus control a large proportion of transactions. Most permissionless blockchain systems tend to use a proof-of-work consensus technique to verify transactions. Two of the most popular permissionless blockchain technologies are Bitcoin and Ethereum, although they are quite different in implementation and have different uses as enumerated below.

- **Bitcoin:** Böhme *et al.* (2015), describes Bitcoin as an "online communication protocol" which has making electronic payments possible as its main purpose while relying on cryptographic validation and use of private cryptographic keys to validate and keep transactions secure. As Khalilov and Levi (2018) points out, the address of Bitcoin users

are essentially hashes of their public key and Bitcoin uses blockchain as a public electronic ledger of all transactions which is shared between all users and verified by them. Hence, the blockchain itself serves as a single source of verification.

- **Ethereum:** According to Vujičić *et al*. (2018) and Pontiveros *et al*. (2018), Ethereum is essentially a programmable blockchain platform upon which it is possible to create personalized ownership rules and transaction formats. It serves as an abstract layer upon which blockchain applications can be built upon. Thus, it can also be seen as a permissioned blockchain especially with it's ability to offer user written Turing complete smart contracts (Dinh *et al*., 2017).

## 2.4 Blockchain structure

As Luo and Huang (2017) mentioned, a block is the "basic fundamental component of a blockchain" and is essentially a compilation of relevant pieces of information. A block in general contains a reference to the block before it, some of the previous transactions and a means of proving its validity by consensus (Kaushal *et al*. 2017). Due to the varying nature of the different types of blockchains available, this also results in their structural details being different (Li *et al*. 2018). In the most basic form though, the fundamental structure of blockchain revolves around hashes.

For instance, in a proof of work based blockchain network, a node X creates a block by solving a cryptographic puzzle (that is, iterating through some hash functions until a certain format is obtained) and then other nodes validate the created block. The timestamping of each transaction in the chain depends on the nature and implementation of the blockchain. Usually, timestamping in blockchains are managed in a decentralized way, for example in the permissionless a hash value is computed for each block created and propagated to all nodes for validation and the hash-value of that timestamp is then included in the timestamp of the next block and digitally signed (Yanga *et al*. 2018). In a situation where 2 blocks are created and published at the same time with both pointing to the same preceding block, this results to a fork, where the blockchain separates into two branches (Liu *et al*. 2017). This is one of the challenges faced in blockchain technologies.

## 2.5 Blockchain security

In general, no practical cryptosytem has been capable of being rigorously shown to have a perfect or unconditional security and this is inherently a core characteristic of any technology which

depends on cryptographic solution. According to Stefan Wolf (1997), "computational security can only exist under certain assumptions", which includes limitations of computing power and difficulty of underlying cryptographic problem to be solved. Blockchain technology itself, has inherent security risks (Dai *et al.*, 2017). The security basically depends on a lot of factors including and not limited to how the network itself is setup, how consensus is obtained within the nodes, how transactions are signed, time-stamped and verified, how the reward systems are setup such as for Bitcoin miners, and of course the physical security itself considering users have private keys which sometimes are not stored securely. In this thesis, the focus will be on the digital signing process, the time stamping techniques employed, and the hash functions utilised in order to provide a fairly deep analysis, as these are core to the technical and cryptographic security of blockchain technologies.

- **Digital signatures:** As pointed out by Wolf (1997), a digital signature scheme is a means of safeguarding integrity of messages, whereby a sender digitally sign messages using a secret key before sending, as this can be verified by the recipients with the sender's shared public key. Similarly, Zhou *et al*. (2006) studies describes digital signatures as a cryptographic evidence which cannot be denied. They are an essential part of most practical security implementations and in general, most of these implementations are based on the difficulty of numerical factoring problems or discrete logarithm mathematical problems. In recent times, there has been implementations utilizing lattice problems to counter quantum computing (Howe *et al*., 2015).
- **Time stamping and Consensus techniques:** According to Satoshi Nakamoto (2009), digital signatures, time stamping, and consensus algorithms are intricately linked. In general, blockchain transactions, such as Bitcoin transactions, are time-stamped by using a hash-based consensus to chain them together making it difficult to change the transaction, and this would then necessitate having to perform the complicated calculations, that is, the hashes for all the chained blocks all over to have a proof of work and somehow still remain the longest chain in the network. Similarly, Mingxiao *et al*. (2017) highlighted the importance of consensus and timestamping as a means of addressing the issues of double spending and colluding nodes. They also identified and described some of the major consensus algorithms, the most common ones being proof-of-work (PoW), proof-of-stake (PoS), practical byzantine fault tolerance and delegated-proof-of-stake (DPoS). As Li *et al*.

(2018) states so simply in their literary work, the key difference between PoS and PoW is that a PoW scheme establishes the credibility of a transaction or data block by utilizing the solution of cryptographic puzzles while PoS schemes largely depend on a proof of ownership for establishing credibility. There is a need for consensus schemes to keep participants honest. A good example being schemes like PoW and DPoS, which propels or motivates participants to avoid bias by creating a high level of competition for the right to produce the next block in the chain.

- **Hash functions and Collisions:** It can be seen from the earlier sections in this writeup how important hashes are to the security of blockchain technology. In the simplest form, a hash function is simply a one-way function which takes an input **A** and transforms it into a fixed length output **B**, that is**,** $\boldsymbol{A \Longrightarrow f\,(A\,) \Longrightarrow B}$, while collisions are simply rare cases where different inputs end up having the same hashed output. According to Chi and Zhu (2017), hashing techniques are basically used in transforming data into shorter fixed length representations, and thus have lots of applications in data storage especially, enabling quick database operations using short compressed hashes that are mapped to the actual data.

  They also suggested hashing techniques could be viewed from two perspectives which are data-oriented hashing techniques which they further broke down into data-dependent and data-independent hashing techniques; and security-oriented hashing techniques which is also broken down into cryptographically secure hashing techniques as used by digital signatures and public cryptosystems, and cryptographically insecure hashing techniques such as those used by domain name servers. In this thesis, one of the main focus is on cryptographically secure hash functions, and these generally have the characteristics of being collision-resistant and preimage resistant that is one way and not possible to get the inverse of the hash image (Chi and Zhu, 2017). Similarly, Gervais *et al*. (2016) explains that a hash-based PoW consensus algorithm involves finding a nonce value which when hashed with a hash of a previous block, produces a value smaller than the target value. Zhou *et al*. (2006) also indicated that the validity of digital signatures is managed using one-way hash-chains.

## 2.6 Blockchain uses

One of the things which so many different authors have tried to show a lot of times, is the fact that blockchain as a technology has had a huge impact in how transactions are being carried out and is

continually shaping it and could become an integral part of lives. There are so many applications of blockchain technology in various industries, which includes but is not limited to the few mentioned below.

- **Smart contracts:** It is extremely difficult to talk about blockchain applications (particularly Ethereum) without mentioning smart contracts. As discussed previously, smart contracts are in a very basic description simply computer codes which enable simple business transactions based on a consensus algorithm (Sato and Himura, 2018; Letourneau and Whelan, 2017). Also, as pointed out by Pontiveros *et al*. (2018), one of the intended goals of smart contracts as in Ethereum, is to have "specialized reusable modules" which can form larger and more complex smart contracts and thus enabling a reduced blockchain size, and a lot of the industrial applications discussed subsequently are built upon the concept of smart contracts.
- **Logistics:** In their work on digital supply chain transformation, Korpela *et al*. (2017) discussed how blockchain provides support to supply chain integration, and the security and effectiveness of supply chain transactions. They identified the application of smart contracts, time-stamping and blockchain ledger as the most important functionalities with which blockchain technology impacts the supply chain industry in addition to the security provided by digital signature and public-key encryption of transactions and data.
- **Internet of things and smart cities:** Banerjee *et al* (2017) discussed about the application of blockchain in sharing of datasets and in self-healing of compromised or damaged IoT devices thus maintaining integrity. Similarly, Sharma and Park (2018) discussed the use of blockchain to maintain security in smart cities which are based on "intelligent, autonomous and distributed network infrastructure".

Some other notable applications include mortgages, electronic voting systems, digital identity and records, securities and many more (Ali *et al.*, 2018).

### 2.7 Blockchain and Quantum computing

According to Howe *et al*. (2015), the emergence of quantum computers poses a huge threat to crypto-systems as we know it, as most crypto-systems are based on difficulty of discrete logarithm problems (DLP) or assumed "hard" mathematical problems such as factoring problems and elliptic curve cryptography; which would be easy to break considering the huge computational power

which quantum computers can achieve. This consequentially would impact every application and technology built upon the current crypto-systems available now, hence the rise of lattice-based computational cryptography.

Lattice-based computational problems are considered to be resilient to quantum reductions or quantum-computer attacks (Howe *et al*. 2015). These includes problems such as shortest vector problems and closest vector problems in the lattice environment. There are already practical implementations of lattice-based public-key encryption and digital signature schemes with lots of research going into lattice-based cryptographic concepts, such as functional encryption, group signatures schemes, identity and attribute-based encryption, and fully homomorphic encryption. In a slightly different perspective, Buchmann *et al*. (2009) compared the security and performance of a few lattice-based encryption schemes showing some example attacks and countermeasures.

## 2.8 Cryptographic security assessment

As expressed by Park and Shin (2017), estimating what is required in the security of an information system or technology is a complicated process as it often involves considering many conflicting criteria and a wide range of factors. Hudic *et al*. (2017) pointed out that having the appropriate metrics is one of the most essential building blocks for making any assessment on an information system asset. Hence, one of the goals of this research work is to understand what is important in blockchain cryptographic security and thus provide an artefact or framework which can be used in assessing the most important areas of consideration in blockchain security.

# 3. METHODOLOGY

## 3.1 Aim of research approach

According to Dresch *et al.* (2015), the research methods serves as a map and gear kit in the quest of solving a given research problem and needs to suitable to the problem. Due to the diverse nature of this particular area of research and its unique nature of still being in the early stages despite being built upon old concepts, there is a need for a qualitative and explorative approach to this research. Thus, this research focused on applying a qualitative approach to identify and understand important considerations to make when assessing a blockchain cryptographic security. Furthermore, the study aims to establish a model that can be adapted to varying contexts, which can then be used in assessing the factors that are important in the security of blockchain technology.

## 3.2 Methodological considerations

The choices made in a research approach and methods, go a long way in having an effect on how a research question is addressed. The relevance of such methods to the research problem and the scientific legitimacy should be considered during selection, and the most useful and effective approach should be selected (Dresch *et al.,* 2015).

As indicated by Creswell (2009), secondary forms of research and data collection can help reduce a scope of a study in proportion to the time and resources available. Thus, one of the main forces driving the of utilizing a desk research approach in answering this study's research question was the desire to develop an effective solution to the research problem in the most cost-efficient way possible, especially considering the topics discussed are all based on older theoretical concepts.

Additionally, this approach encourages the exploration of existing frameworks and hence building upon frameworks already established in literature.

## 3.3 Research methods

In related literary works, Gervais *et al*. (2016) extended Markov decision process to create a mathematical model, in other words a quantitative framework for analysing security and performance of consensus networks, while Bauspiess and Damm (1992), as well as Damaj and Kasbah (2018), qualitatively defined frameworks describing technical requirements which includes functional and security requirements for hash functions. While these provided a blueprint relevant and complementary to this research, there was a need to combine the desk research

approach to a case study methodology considering that there are so many applications of blockchain, with varying differences in their structural organization. Hence, selecting two case which sufficiently addresses major differences, helps provide a construct and overall case narrative which could be extended to other blockchain applications.

As postulated by Stewart (2012), one of the drivers of multiple case studies, is the possibility of discerning themes or success factors and failure factors common to the cases which can be generalized, and this complements the aims of this study.

Similarly, according to Andersen and Kragh (2010), case studies could be used in a theory-building process to formulate, categorize and organize relationships between the cases being observed.

The two case studies selected for this research were Bitcoin and Ethereum, as these were the most popularly used cryptocurrencies with some amount of documentation available describing their functionality. The analysis of their inner structure, operations and algorithms was designed to represent the relationship between hash function implementations and blockchain operations, thus contributing to the creation of a generic theoretic model which can be tweaked according to the situation and context of use.

# 4. BLOCKCHAIN STRUCTURAL ANALYSIS

## 4.1 Blockchain structure

This chapter is aimed at providing an understanding to the inner workings of a blockchain. As stated in the chapter two, a blockchain is simply a distributed chain of data blocks in which each block is linked cryptographically to the block preceding it as illustrated in Figure 2.

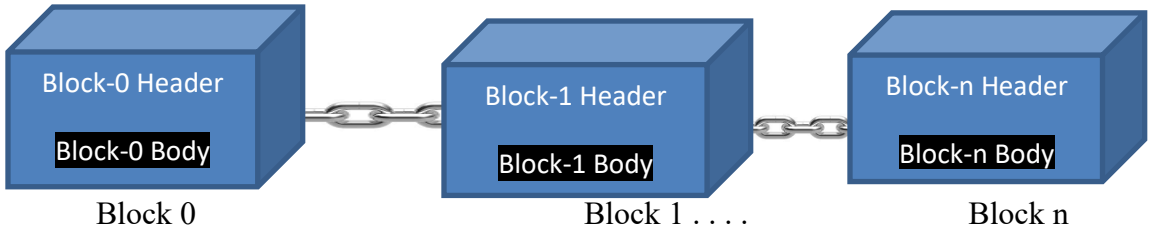


**Figure 2:** A Basic Blockchain

In a basic form, blockchain as a technology is quite similar to having this thesis work for instance as the blockchain, copied onto some group of friends' laptops, such that the pages which would be the blocks in each copy, and are linked together with a mathematical function. To add a new page to the thesis, one friend has to solve a certain riddle. If the friend is able to solve a particular riddle, he or she then broadcasts the new page chained to the previous pages, and if there are no longer chains of messages from other users already broadcasted and verified, then the user's chain stands a chance of becoming the chain to be followed.

According to Zheng *et al*. (2017), a blockchain is just a sequence of blocks, such that each block has the hash of the previous block header contained within its header as well as a merkle-tree root hash, while the transactions in other words pending transactions and confirmed transactions of the last block are contained in its body. Though for some blockchain, such as Ethereum, the hash of the subsequent block in the sequence is also contained in the header. Generally, the first block is usually called the genesis block, and the transactions it contains are hashed according to a merkle tree structure, creating one Root merkle-tree hash which is included in its head to be passed to the following block (Yaga *et al*., 2018).

When a new user joins a blockchain network, it receives a full copy of the blockchain and its transactions, which means all the blocks created with their validated transactions will be copied

into the user's system, and these blocks can be confirmed using the hashes, in other words the merkle-tree root hash and preceding block header hashes.

### 4.2 Blocks

In general, a block usually has the following components as shown in Figure 3.

- A timestamp
- A hash value for the previous block's header
- A merkle-tree root hash (ie a hash which is the summary of all transactions in the block)
- A nonce value (used in solving the hash puzzle)
- The current block hash value
- The size of the block
- The block version
- The block's number
- The set of transactions contained in the block

The header contains at least the first four components (timestamp, hash values for previous block header and merkle-tree root hash, and nonce value).

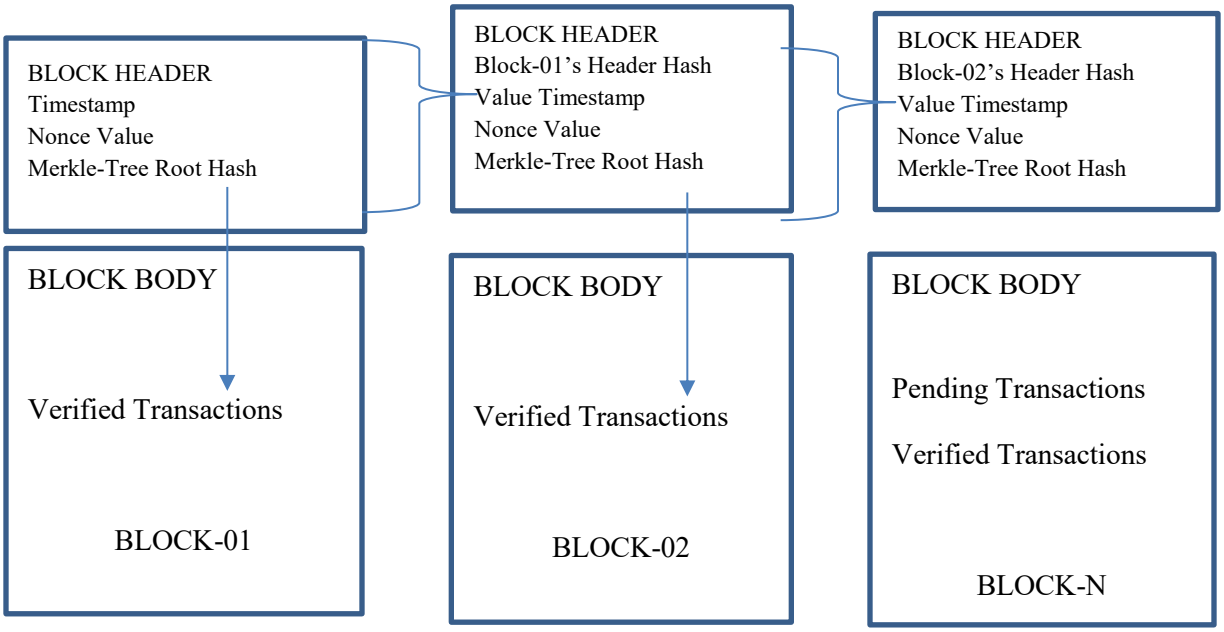


**Figure 3:** Block Header and Block Body Contents

As illustrated in Figure 2, the verified transactions or data in a block are hashed into one single merkle root hash which forms part of the block header as well as the hash value of the previous block, and the contents of the block header are then further hashed into a single value which will be contained in the next block header. Essentially, the entire block is summarized into one single hash (Figure 4), which means any changes to the transactions would immediately make the hash value in the next block header to be invalid, and changes rejected-this helps in providing immutability to blockchains.

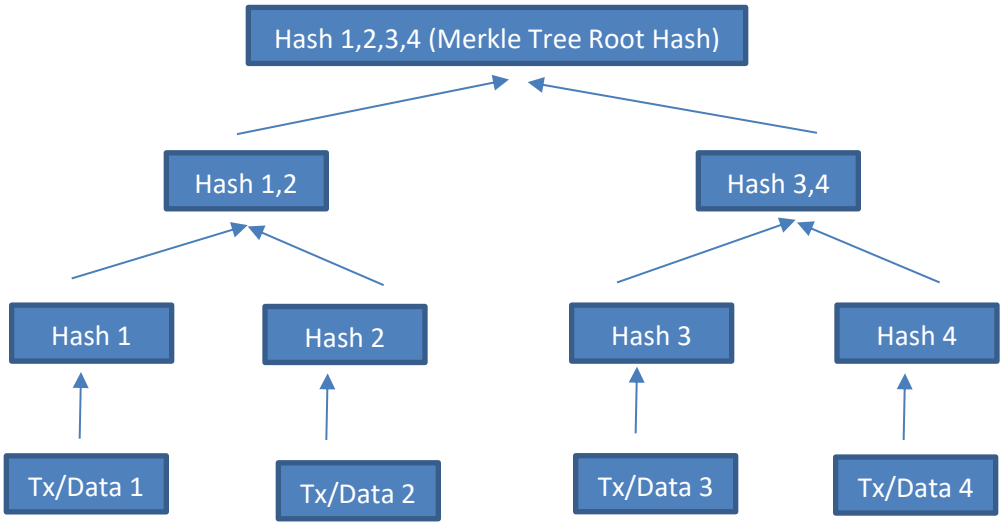


**Figure 4:** Merkle Tree Structure for Data hashing for a block

### 4.3 Blockchain transactions

In the blockchain, new pending transactions which are the candidate transactions, are broadcasted to all nodes in the network as depicted in Figure 5. They wait in a transaction queue and if they are accepted according to the consensus requirements of the network, then they will be added as verified transactions in a block and the updated block or new block is published and distributed to all other nodes on the network. In other words, the first node which is able to solve the cryptographic challenge for a new transaction according to the nonce value gets to add the transaction to a block and publish the block. Then all other nodes will independently verify the validity of the transactions in the broadcasted block and update their ledgers. If there are any discrepancies or invalid transactions, the block would be rejected, else it would be accepted and added to the network.

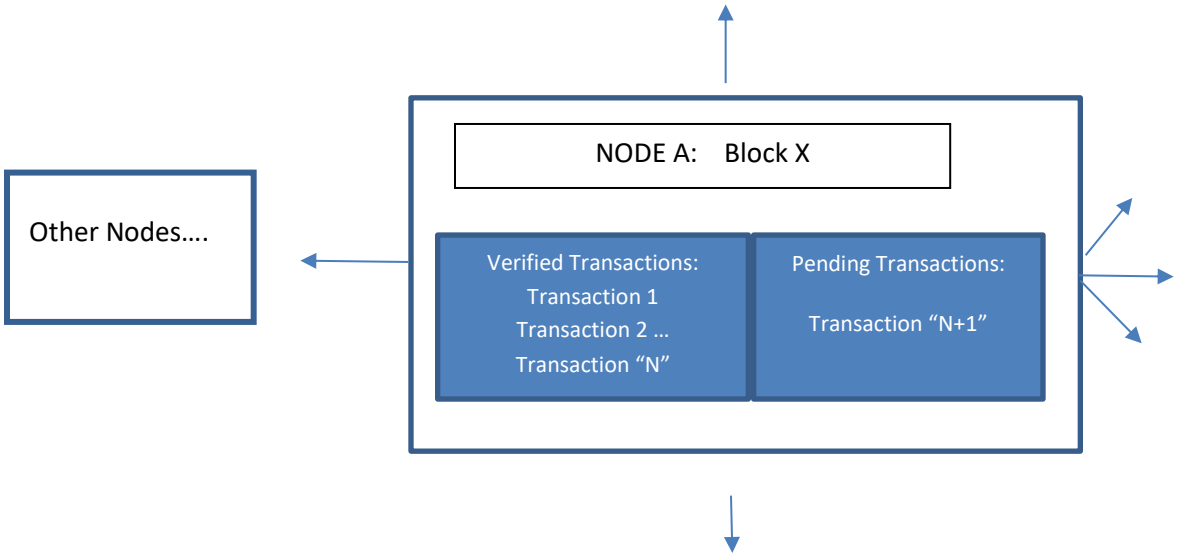


**Figure 5:** "Node A" broadcasting a candidate transaction ("N+1") to other nodes for validation

All other nodes would abandon their previous attempts to solving the cryptographic puzzle, and subsequently resume efforts on adding another block to the new chain which was broadcasted. If two nodes happen to broadcast at the same time, a "fork" is established in the chain, and the nodes in the blockchain network will then proceed with the chain which eventually has the longest chain (larger amounts of blocks) and disregard the chain which becomes shorter.

Each transaction is digitally signed (signed with private keys) using an asymmetric-key cryptography usually elliptic curve digital signature algorithm, and hence can be verified utilizing corresponding public key this has a huge impact on the idea of accountability in blockchains (Zheng *et al*., 2017; Yaga *et al*., 2018).

### 4.3.1 Immutability

As mentioned above, any changes in a block, even the slightest change in a single bit would necessitate changing all the subsequent blocks, and this has to be done at a faster rate than the

entire network combined, which stretches the limits of possibility. This is one of the main advantages offered by blockchain technologies and is made possible by hashing.

In a basic form, how hashing works is that in 512 SHA for example, a hash of the number "*7*" is:

> **"F05210C5B4263F0EC4C3995BDAB458D81D3953F354A9109520F159DB1E8800BCD45B97C56DCE90A1FC27AB03E0B8A9AF8673747023C406299374116D6F966981**";

While that of "*Cryptography is an interesting subject – account number 0000011149 etc@5*" would be:

> "**A8EBFA6CA38D48A08DF834C143C3D6F7218A9FC85BF30DC3607CD468227A9F77BF532844BA66218DEF74B26FC8134DFDC297A66C504C27ED93062F5DE05AF4FF**".

Both hashes are of a fixed size and length, and the same strings will continue producing the same hashes, so it is easy to verify but difficult to reverse the hash back to obtain the strings especially because it is of a fixed length with either some compressions or paddings. This importantly makes accessing blocks or data operation on blocks possible in almost a linear time complexity of $\boldsymbol{\theta}(\boldsymbol{n})$ with respect to the content length (Chi and Zhu, 2017).

The idea of immutability of blockchain stems from the fact there exists a consensus among all nodes on what data exists in the blockchain, which means each node in the blockchain has the same copy of blocks. Hence for a malicious user to change the contents of a block, he would have to compute and create more blocks than the longest chain already existing in the blockchain. This means having to compete against and outpace the rest of the network, which in practice is not feasible. This implies that the deeper a block is in a network, the harder it is the modify its content or reverse its transactions, which is why it is recommended to wait for at least three blocks (three confirmations) to ensure non-repudiation in PoW networks, and some cryptocurrency exchanges insist on six confirmations.

### 4.3.2 Accountability

The basic concept of accountability in blockchains revolves around the idea of immutability and digital signatures, as well as the notion that mostly transactions are linked to a particular node or address on the network although the exact details of the identity might be hidden. It is important to mention though that blockchain solutions such as Monero and Zerocoin add extra layers of

obfuscations and anonymity; and avoid double spending. Zerocoin for example implement zero-knowledge-proof-based validation of transactions, without having to expose any information about the transaction.

The concept of accountability can be explored from the perspective of transactions and the perspective of regulations as enumerated below respectively.

- As previously mentioned, blockchain technologies are usually built upon asymmetric-key cryptography and transactions are designed towards immutability, and this means for any transaction to go through, it would have been digitally signed with a private key that is not known to the public and can be verified with the corresponding public key. The transaction is practically indelible; hence to delete or modify it would have to involve changing all subsequent blocks in the blockchain. Consequently, in private blockchains for instance every node can be held accountable to all transactions they make. These transactions are transparent and the blockchain provides a single version of the truth across the whole network.
- From the perspective of regulations, one recent and arguably most important regulation affecting the use and creation of data is the General Data Protection Regulation (GDPR), and research work such as that of Neisse *et al*. (2017), suggests that smart contracts can be utilised in ensuring transparency and access of data. In other words, organizations can be held accountable to usage of data, through the implementation of auditable smart contracts.

### 4.3.3 Privacy and Anonymity

It seems strange talking about transparency and privacy or anonymity simultaneously, but regardless this is one key feature blockchain offers. In most cases especially in cryptocurrencies, the public addresses of nodes are usually a hashed value of the public keys, which do not reveal the true identity details of the node to all on the network. This allows nodes to be entirely transparent in the transactions while maintaining their anonymity and/or their privacy in terms of their identity.

As discussed by Khalilov and Levi (2018), anonymity can be regarded as hiding “who”, in other words the details of the owner of a transaction while privacy can be seen from a contextual viewpoint, that is, hiding the context of the transaction. For example, when nodes have a hashed

value of the data in a transaction, it is transparent to all nodes that the transaction took place, and which public address the transaction is linked to. For instance, the digital address or name of the node on the network can be seen, but not the exact context in which the transaction was carried out nor the actual details of who made the transaction in the real world. This explains why cryptocurrencies have been known to be used for illicit purposes in the dark world such as the darknet market "silk road" (Mehta *et al*., 2017). Though cryptocurrencies do not provide a complete anonymity, considering the IP addresses can be stored as meta data as well as time of transactions, and these transaction logs can lead to discovering identities when analysed holistically.

### 4.3.4 Scalability

Having discussed some of the merits of using blockchains, now it is imperative to also discuss some of the demerits. Scalability is one major challenge which blockchain technologies face. For instance, when it comes to cryptocurrencies based on proof-of-work consensus systems, the amount of energy required keeps increasing as the difficulty levels of cryptographic puzzles increases, such that it is bound to become unstainable at some point. In some cases, the amount of energy required to power one mining-pool (that is, a pool of nodes, mining and creating cryptocurrencies) is more than enough to power a small city or small countries (Vranken Harald, 2017). Likewise, as it takes 10 minutes to mine one block in a bitcoin blockchain for example and 150 seconds in Litecoin, the overall delay on the entire network could become exponential, the larger the chain becomes, and also the more data it contains.

In other words, the difficulty and requirements for bandwidth, computing power, storage memory, asymmetric-key management, and network complexity management all increases as the blockchain network increases. This will result to so many difficulties when utilized in an internet of things context with thousands of data transactions every second. To put it simply, one has to make a trade-off between scalability, security and decentralization as the blockchain increases due to performance optimization issues. There have been proposed solutions such as sharding which involves breaking blocks into pieces and sharing it across the network during the verification process, but these are still works in progress.

### 4.4 Forking and Double spending

One other prevalent issue which comes with decentralized blockchain is forking, and this is simply a situation where two or more nodes publish a new block in a blockchain pointing to the same preceding block simultaneously. For example, if node X, node Y and node Z each publish a block (block X2, block Y2, block Z2) at the same exact moment, and the 3 blocks all have block A1 (from node A for example) as their preceding block, then the blockchain at that moment has 3 different forks from the penultimate block A1. In other words, 3 legitimate chains exist, and other nodes can add blocks to any of the chains. This obviously constitutes a huge issue as the transactions in the different chain forks are contradictory. Usually, how this is dealt with in most blockchains especially cryptocurrencies, is that nodes add blocks to only the longest of the chains. This means adding blocks to the fork which increases the fastest and has the chain with the longest series of blocks agreed by consensus. All other forks are rejected and cut off, and subsequently the nodes with the removed forks can resubmit their transactions to a transaction pool waiting to get added to a new block in the valid chain (Liu *et al*., 2017). In Bitcoin, in order to avoid the entire blockchain becoming invalidated in an attack, the use of checkpoints was introduced which freezes the blockchain from the genesis block up to a particular predefined block (Alqassem and Svetinovic, 2014).

There is a direct relationship between forking and double spending, as malicious users can use the possibility of forks in coordinating double spending attacks in cryptocurrencies (Anceaume *et al*., 2016). Double spending basically is a phenomenon that occurs in a cryptocurrency when the same coin is effectively utilised in two different transactions. According to Mehta *et al*. (2017), during a double spending attack, an attacker can create two transactions, one bearing the address of a merchant or service provider as the output address and the other transaction bearing the attackers address (which is not broadcasted yet), with the intention of convincing the merchant to accept the transaction before publishing and ensuring the other transaction is confirmed and added to the blockchain by other nodes. Attackers could also simply just try creating two blockchains (a fork), and this basically works in the attacker's favour if the illegitimate transaction becomes the approved longer chain. In their work, Natoli and Gramoli (2017) demonstrated a 94% success rate in a double spending attack in a network simulation where they are able to disrupt communication between groups with a similar mining power.

# 5. OVERVIEW OF BLOCKCHAIN CRYPTOGRAPHY

## 5.1 Digital signatures

According to Zheng *et al.* (2017), digital signing works in such a manner that every node has a private and public key. The private key is used in signing transactions and kept secret, then these digitally signed transactions are broadcasted over the network and they can be easily verified using the available public keys. One of the most commonly mathematical algorithms used in digital signatures is the elliptic curve digital signature algorithm (ECDSA).

### 5.1.1 Elliptic curve digital signature algorithm

The ECDSA can be broken down into 3 main activities which includes the key generation, signature generation and signature verification. These activities are enumerated below.

---

Step 1: Key generation

---

Essentially, the outputs of the key generation are the public key $\boldsymbol{K_{pub}}$ and private key $\boldsymbol{K_{pr}}$, and to generate this, the node or user who wishes to sign a transaction will need;

- an elliptic curve **E** defined by $\mathbf{y^2} = \boldsymbol{x^3} + \mathbf{a}\boldsymbol{x} + \mathbf{b}$ over a prime finite field $\mathbf{F}_{\boldsymbol{p}}$ with modulus $\boldsymbol{p}$, the coefficients $\boldsymbol{a}$ and $\boldsymbol{b}$, and a point $\boldsymbol{A} = (x_A, y_A)$ which can generate a cyclic group of order $\boldsymbol{q}$ (where $\boldsymbol{q}$ is a prime)
- to choose a private key $\boldsymbol{d}$ from the integer interval $[\,\mathbf{1},\ \boldsymbol{q}-\mathbf{1}\,]$
- and then calculate a new point $\boldsymbol{B} = (x_B, y_B)$, where $\boldsymbol{B} = \boldsymbol{d}\bullet\boldsymbol{A}$

The public key then becomes the set $\boldsymbol{K_{pub}} = (\boldsymbol{p}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{q}, \boldsymbol{A}, \boldsymbol{B})$ while the private key is $\boldsymbol{K_{pr}} = \boldsymbol{k}$

---

Step 2: Signature generation

---

Having created the keys, the digital signature itself is the pair of values ***(r, s)*** where $\boldsymbol{r}$ and $\boldsymbol{s}$ are computed utilizing the point $\boldsymbol{A} = (x_A, y_A)$ created above, a new point $\boldsymbol{R} = (x_R, y_R)$ together with a temporary key $\boldsymbol{K_E}$, and the transaction $\boldsymbol{M}$ to be signed and its hash in the following steps:

- calculating the Hash of the transaction $m_H$ = HASH($\boldsymbol{M}$)
- finding $\boldsymbol{z}$ the leftmost bits of the hashed transaction $m_H$ with bit length $\boldsymbol{q}$

- choosing a temporary key $\boldsymbol{K_E}$ from the integer interval [ $\mathbf{1},\ \boldsymbol{q}-\mathbf{1}$ ]
- computing the new point $R = K_E \times A$ (that is R = $(x_R, y_R)$ )
- The first signature value $\boldsymbol{r} = x_R$ (mod q) such that $x_R \neq 0$, if $x_R = 0$, a new temporary key $\boldsymbol{K_E}$ is chosen.
- The second signature value $s \equiv (z + d \times r)K_E^{-1}\ mod\ q$; such that $\boldsymbol{s} \neq 0$, if $\boldsymbol{s} = 0$, a new temporary key $\boldsymbol{K_E}$ is chosen.
- Subsequently, the public signature pair becomes ***(r, s)***

---

Step 3: Key verification

---

To verify the signature, the receiving node needs the public key set $\boldsymbol{K_{pub}}$ = (***p***, ***a***, ***b***, ***q***, ***A***, ***B***), two auxiliary values $\boldsymbol{u_1}$ and $\boldsymbol{u_2}$, the pair ***(r, s)***, the inverse of ***s***, that is $\boldsymbol{s^{-1}}$, the hashed message or transaction ***z***, and a new point ***J*** that is ***J*** = $(x_j, y_j)$, and these are calculated in the following steps.

- calculating the inverse of ***s***, that is $\boldsymbol{w \equiv s^{-1}}$ (mod q); this means $\boldsymbol{s \bullet s^{-1}} \equiv 1$ (mod q)
- calculating a value $u_1 \equiv w \times z$ (mod q)
- calculating a value $u_2 \equiv w \times r$ (mod q)
- calculating a new point $J = u_1 \times A\ +\ u_2 \times B$ (that is ***J*** = $(x_j, y_j)$ )
- Then the signature can easily be verified by simply checking if $\boldsymbol{x_j}$ is congruent to ***r*** (mod ***q***) or not. If it is congruent, then the signature is valid.

---

The details of the steps above will not be proved in this literary work, but the following very simple example illustrates the concept explained.

Assuming the elliptic curve $y^2 = x^3 + 2x + 2\ \ /F_{17}$ is chosen and a point ***A*** = (5,1) which generates a cyclic group of order 19, such that **p** = 17, **a** = 2, **b** = 2, and order **q** = 19. It is trivial to check and see that the elliptic curve satisfies the two conditions given in section 5.1.1. If Alice wants to sign a transaction which has a Hash value of 41, she generates the key set as follows.

- She first selects a private secret key **d** = 11;
- computes point $B = d \times A = 11 \times (5,1) = (13,10)$, and makes $(p, a, b, q, A, B)$ public

Then she proceeds to generate the signature thus:

- selects a temporary key $\boldsymbol{K_E}$ = 8

- and computes the point $R = K_E \times A$ = 8×(5, 1) = (13, 7)
- this implies that **r** = $x_R$ = 13.
- Then she computes $s \equiv (z + d \times r)K_E^{-1} \, mod \, q \equiv (41 + 11\times13) \times 12 \equiv 4 \text{ mod } 19$; signature pair **(r, s)** = (13, 4)

Bob the receiver can then verify the transaction thus:

- He computes **w** the inverse of **s** (mod **q**); $\mathbf{w} \equiv \mathbf{4}^{-1} \text{ (mod 19)} \equiv 5$
- Then he computes the two auxiliary values $u_1 \equiv w \times z \text{ (mod 19)} \equiv 5 \times 41 \equiv 205 \equiv 15$
- and $u_2 \equiv w \times r \text{ (mod 19)} \equiv 5 \times 13 \equiv 65 \equiv 8$;
- and the new point **J** = $u_1 \times$**A** + $u_2 \times$**B**; = 15×(5,1) + 8×(13,10) = (3,16) + (0,11) = (**13, 7**)
- therefore, Bob can validate the signature since $x_j$ is congruent to $\boldsymbol{r} \equiv 13$

## 5.2 Hash functions and Collisions

As explained in chapter four of this work, hash functions form an integral part of blockchain transactions. Consequently, the hashing process will be examined closely in this section. A hash function is simply a function which receives a message that is an arbitrary length of string as input and outputs a fixed length bit string, this output is usually called the message digest or hash value.

**Theorem 5.1** According to Smart (2016), a practical cryptographic hash function is required to have three main properties as highlighted below.

- Preimage resistance: given a hash value "**y**", it should be computationally hard (a mathematical hard problem) to find the initial message "***x***" from it such that

$$\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{y}$$

- Second preimage resistance: given an input *x*, it should be computationally hard to find a different input $\boldsymbol{x}^{|}$ such that

$$\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{H}\ (\boldsymbol{x}^{|})$$

- Collision resistance: It should be computationally hard to find any two distinct messages with the same hash value. Thus, collision resistance implies second preimage resistance, but it does not guarantee that the function is a one-way function (preimage resistance)

$$\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{H}(\boldsymbol{y})\ ;\ \boldsymbol{where}\ \ \boldsymbol{x} \neq \boldsymbol{y}$$

From the properties above, it can be seen that a collision is simply when two different elements in a particular domain map to the same element in a codomain. It is usually assumed the domain is larger than the codomain (Smart, 2016).

According to Yaga, Dylan *et al*. (2018), one of the most common hashing algorithms utilized by blockchain technologies is the secure hash algorithm (SHA) which has various families which includes SHA-0, SHA-1, SHA-2 and SHA-3. The SHA-0 (which was withdrawn and replaced with SHA-1), SHA-1, and SHA-2 families are based on a Merkle-Damgård's construction of a hash function while the SHA-3 is based on a concept called sponge construction of a hash function. The members of each family are characterized by many factors such as their number of rounds, their internal state size, the block size they process, and the bit size of their message digest, in other words bit size of hash values and initial values used. For example, the SHA-256 has a 256 bits digest size and SHA-512 has 512 bits digest size. Also, SHA truncated versions exist in SHA 2 family and others. In their work, Dobraunig *et al*., (2015) demonstrated practical collisions for 27 rounds of SHA-512/224, SHA-512/226 and SHA-512 utilizing search tools and differential cryptanalysis. They also assert that these two truncated versions of SHA-512 perform faster while maintaining same security level and hash digest size as SHA-224 and SHA-226.

In this thesis, the SHA-2 (specifically the SHA-512 algorithm) and SHA-3 families and their hash process will be briefly discussed though not extensively. A more detailed description of both can be found in the National Institute of Standards and Technology's (2015) Secure Hash Standard publication.

### 5.2.1 SHA-512 algorithm

Before introducing SHA-512 in this work, a few symbols and notations will be defined – these will be used in describing the hash function.

- $H^{(i)}$ is the $i^{th}$ hash value while $H_j^{(i)}$ is the $j^{th}$ word of the $i^{th}$ hash value, where while $H_0^{(1)}$ is the leftmost word of the second hash value
- similarly, $M^{(i)}$ this $i^{th}$ message block while $M_j^{(i)}$ is the $j^{th}$ word of that $i^{th}$ message block
- $N$ is the number of blocks in a padded message
- $\ell$ is the bit length of the message, while $m$ is the bit length of a message block $M^{(i)}$
- $w$ is the bit length of a word, while $W_t$ is the $t^{th}$ $w$ -bit word

- $K_t$ is the constant value used for each iteration $t$ in the computation of the hash digest
- Then the following logical operations; $\wedge$ (bitwise *AND* operation), $\vee$ (bitwise *OR* operation), $\neg$ (bitwise "*complement*" operation), + (addition modulo $2^w$), >> (right-shift operation), << (left-shift operation), $\oplus$ (bitwise *XOR* operation)
- Lastly the following algorithmic operations;
  - $ROTL^n(x)$ which is the rotate left operation that is $(\mathbf{x} << \mathbf{n}) \vee (\mathbf{x} >> \mathbf{w - n})$
  - $ROTR^n(x)$ which is the rotate right operation that is $(\mathbf{x} >> \mathbf{n}) \vee (\mathbf{x} << \mathbf{w - n})$
  - and the $SHR^n(x)$ which is simply the right-shift operation $(\mathbf{x} >> \mathbf{n})$
    where **x** is a **w**-bit word and **n** is an integer $0 \leq \boldsymbol{n} < \boldsymbol{w}$. Also, the $ROTL^n(x)$is equivalent to $ROTR^{w-n}(x)$ and similarly $ROTR^n(x)$is equivalent to $ROTL^{w-n}(x)$.

The entire algorithm utilizes six logical functions which combines some of the operations defined earlier and is performed on 64-bit words (for example **x**, **y**, and **z**). The six functions are as follows.

- $Ch(\mathbf{x}, \mathbf{y}, \mathbf{z}) = (x \wedge y) \oplus (\neg x \wedge z)$
- $Maj(\mathbf{x}, \mathbf{y}, \text{and } \mathbf{z}) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$
- $\Sigma_0^{\{512\}} = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x)$
- $\Sigma_1^{\{512\}} = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x)$
- $\sigma_0^{\{512\}} = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x)$
- $\sigma_1^{\{512\}} = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x)$

The algorithm also uses a sequence of eighty 64-bit words as $K_t$ which are $K_0^{\{512\}}, K_1^{\{512\}}, \ldots, K_{79}^{\{512\}}$. For example, the first constant in hexadecimal form is $K_0^{\{512\}}$ = 428a2f98d728ae22.

The SHA-512 algorithm can be basically broken down into two main stages: the pre-processing stage and the hash computation stage.

**The pre-processing stage** involves padding the message to be hashed, breaking it into blocks subsequently and then setting an initial hash value. For a message **M** of $\boldsymbol{\ell}$ bits;

- First, "**1**" is appended to the end of **M**
- then **k** zero bits are also appended such that $\boldsymbol{\ell} + 1 + \boldsymbol{k} = 896 \bmod (1024)$ that is $\boldsymbol{k} = 896 - (1 + \boldsymbol{\ell})$

- then a 128-bit block representation of l is also appended (e.g. 000000… $\ell$), thus making the entire padded message to sum up to 1024-bit

For example, to pad a message containing just the string "**UCHE***",* where ASCII 8-bit values of the characters are U = 01010101, C = 01100011, H = 01101000, and E = 01100101. Hence, the length of the message here is 8x4 (4 ASCII letters that is 4 bytes) = 32bits, and 1 is appended to the end of the message followed by a padding of 896 – (32+1) = 863 zero bits, and lastly a 128bit expression of $\ell = 32$ as follows;

| "**U**" | "**C**" | "**H**" | "**E**" | 871 zero-bits | $\ell = 32$(that is 128 bits) |
|---|---|---|---|---|---|
| 01010101 | 01100011 | 01101000 | 01100101 | 1 00…00 | 00…0100000 |

What this achieves is to simply transform the message block a multiple of 1024 bits, which enables subsequently parsing the entire padded message into *N* number of 1024-bit blocks ($M^{(1)}$, $M^{(2)}$,…, $M^{(N)}$). Each block being 1024-bit in size which means each block is composed of sixteen 64-bit words ($M_0^{(i)}$, $M_1^{(i)}$, …, $M_{15}^{(i)}$).

**The hash computation stage** involves using a set of predefined eight 64-bit words ($H_0^{(0)}$, $H_1^{(0)}$,…, $H_7^{(0)}$) which are calculated using the first 64-bits of the fractional part of the square roots of the first eight prime numbers (e.g. $H_0^{(0)}$ = 6a09e667f3bcc908). These initial hash values are used in computing the intermediate hash values. The algorithm is described below.

Each message block $M^{(i)}$ for $i = 1$ to $N$ (that is $M^{(1)}$, $M^{(2)}$, …, $M^{(N)}$), is processed in four basic steps – message schedule preparation; initialization of working variables used in permutating; permutation and mixing; and computation of intermediate hash values. These steps are carried out as enumerated below.

---

Step 1: Message schedule preparation

---

- $W_t = M_t^{(i)}$ for $0 \leq t < 15$, and
- $W_t = \sigma_1^{\{512\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{512\}}(W_{t-15}) + W_{t-16}$ for $16 \leq t \leq 79$

This means the first sixteen words are the words contained in the input array, and the next sixty-four words are successively calculated utilizing the function σ (which is a combination of the different rotate functions).

Step 2: Initialization of permutation variables

The variables used in the permutation and mixing (a, b, c, d, e, f, g, and h) are initialized using the following hash values:

- $a = H_0^{(i-1)}$, $b = H_1^{(i-1)}$, $c = H_2^{(i-1)}$, $d = H_3^{(i-1)}$, $e = H_4^{(i-1)}$, $f = H_5^{(i-1)}$, $g = H_6^{(i-1)}$, $h = H_7^{(i-1)}$.

Step 3: Permutation and mixing

The variables, input words and hash values are mixed up and permutated using the functions **Ch()**, **Maj()** and Σ for t = 0 to 79.

- $\mathbf{T_1} = \mathbf{h} + \Sigma_1^{\{512\}}(e) + \mathbf{Ch}(e, f, g) + K_t^{\{512\}} + W_t$
- $\mathbf{T_2} = \Sigma_0^{\{512\}}(a) + \mathbf{Maj}(a, b, c)$
- $h = g$, $g = f$, $f = e$, $e = d + T_1$, $d = c$, $c = b$, $b = a$, $a = T_1 + T_2$.

Step 4: Computation of intermediate hash values

Lastly, the $i^{th}$ intermediate hash values are then calculated

- $H_0^{(i)} = a + H_0^{(i-1)}$, $H_1^{(i)} = b + H_1^{(i-1)}$, $H_2^{(i)} = c + H_2^{(i-1)}$, $H_3^{(i)} = d + H_3^{(i-1)}$, $H_4^{(i)} = e + H_4^{(i-1)}$, $H_5^{(i)} = f + H_5^{(i-1)}$, $H_6^{(i)} = g + H_6^{(i-1)}$, $H_7^{(i)} = h + H_7^{(i-1)}$.

These four hash computation steps are repeated $N$ -times, which is after $M^{(N)}$ has been processed. Essentially, the 512-bit message input is expanded into a 1024-bit padded message (64-bits array), and this is then permutated and compressed into 512-bits. The output thereafter becomes a hash digest of 512-bit size (a concatenation of the eight 64-bits $H^{(N)}$ hash values (that is $H = H_0^{(N)}H_1^{(N)}H_2^{(N)}H_3^{(N)}H_4^{(N)}H_5^{(N)}H_6^{(N)}H_7^{(N)}$).

### 5.2.2 SHA3-512 algorithm

As mentioned in the previous section, the SHA-1 and SHA-2 families are quite similar, but SHA-3 is based on a quite different concept – the sponge construction and KECCAK permutations. This section will be focused on describing the SHA3-512 in a shallow manner. Unlike the previous section, the SHA3-512 algorithm will not be described here, instead the goal is to briefly explain

what the main set of operations (θ, ρ, π, χ, ι) aim to achieve and point out the key concepts and major differences to SHA-2. The National Institute of Standards and Technology's (2015) Secure Hash Standard publication provides a detailed and lengthy description of this algorithm. Though just as in the previous section, a few symbols and notations utilized in this hash function will be defined first.

- *M* is the message or input to the SHA-3 function while *N* is the input to the sponge function.
- *w* is the lane size (bits) of a KECCAK permutation, while $\boldsymbol{\ell}$ is the binary logarithm of a lane's size *w.*
- Width *b* is the fixed length of inputs and outputs of a function in the sponge construction
- Rate *r* is the number of bits processed from an input or output bits generated when a function is called in the sponge construction.
- Capacity *c* is the subtraction of the rate from the width.
- State is a three-dimensional array of bits which is updated during the KECCAK-p permutation and is usually a 5 by 5 by w array of bits which then consists of planes, slices, sheets, lanes, columns, and rows (Figure 6).
- A[x,y,z] is a bit in the state array that matches the given triple (x, y, z).
- θ ρ π χ ι are the five components or set of operations which a round of permutation is composed of.
- KECCAK[c] is a family of sponge construction functions which uses KECCAK-p permutation as the underlying function and Pad10*1 as the padding rule defined for the KECCAK algorithm.
- SPONGE[f, pad, r] is the sponge function which utilizes an underlying function ***f***, a padding rule ***pad***, and a rate ***r.***

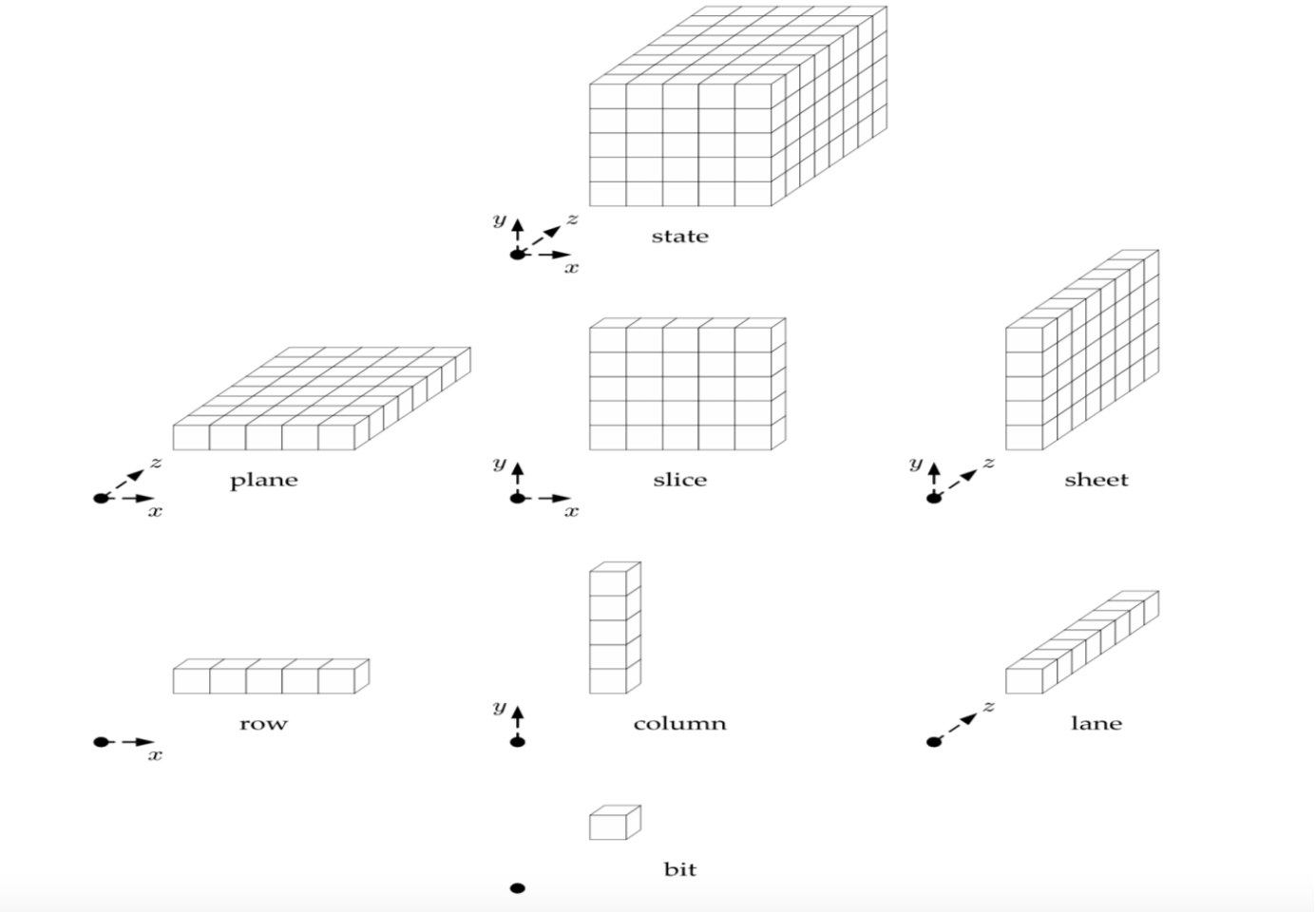


**Figure 6:** Structure of SHA3 "5 by 5 by w" state array of bits

Source: (https://www.cryptologie.net/article/386/)

The SHA3-512 algorithm basically involves converting input strings into 3-dimensional "5 by 5 by *w*" state arrays, and then converting this state arrays back into strings after they have gone through some rounds of internal transformations and permutations. One single round of KECCAK-p permutation is composed of 5 major sequential transformation operations in which the states are repeatedly updated internally as shown in Figure 7.

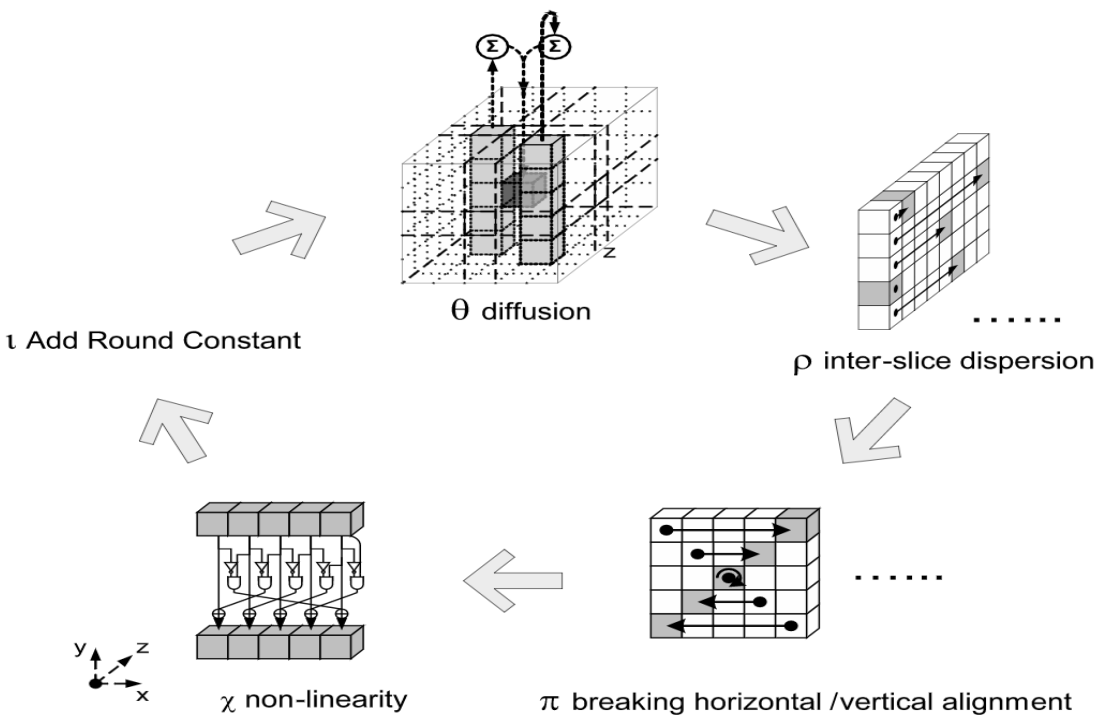


**Figure 7:** (θ, ρ, π, χ, ι) Round Transformations

Source: (https://cryptologie.net/article/387/)

These operations are formally called step mappings and are the following; (θ, ρ, π, χ, ι).

- θ transformation: This receives a state array and XOR every bit in that state with the parities (the XOR summation of every bit in a column) of two columns from the array.
- ρ transformation: This basically rotates the bit contents of every lane by an offset (which depends on the lane's fixed x and y coordinates).
- π transformation: This rearranges and reorders the positions of lanes in the array.
- χ transformation: This basically produces the effect of a linear feedback register, in the sense that it XOR each bit in the array non-linearly with some other two bits in its row.
- ι transformation: This is the only operation of the five that requires another input aside the state array, it also requires a value called the round-constant, which influences how some bits in a given lane will be modified.

The idea is to take an input array, mix it up in a very complicated manner and then return an output string after all permutations, quite similar to how AES for example functions.

The **KECCAK-p**[**b, n**$_r$] transformation essentially receives a string which it converts to a state array **A** and applies these five transformational operations in the order $(\iota\,(\chi\,(\pi\,(\rho\,(\theta\,(\boldsymbol{A}\,)\,)\,)\,)\,), i_r\,)\,)$. This is iterated $\boldsymbol{n_r}$ number of times, where $i_r$ ranges from $(12 + 2\boldsymbol{\ell} - \boldsymbol{n_r})$ to $(12 + 2\boldsymbol{\ell} - 1)$. Then the resulting output is converted back to a string **S**.

On the other hand, the sponge construction basically takes the bit string *N* from the KECCAK permutation and a bit length ***d***, pads it, then subsequently starts squeezing out output bits from its state, hence the name "sponge" as illustrated in Figure 8.

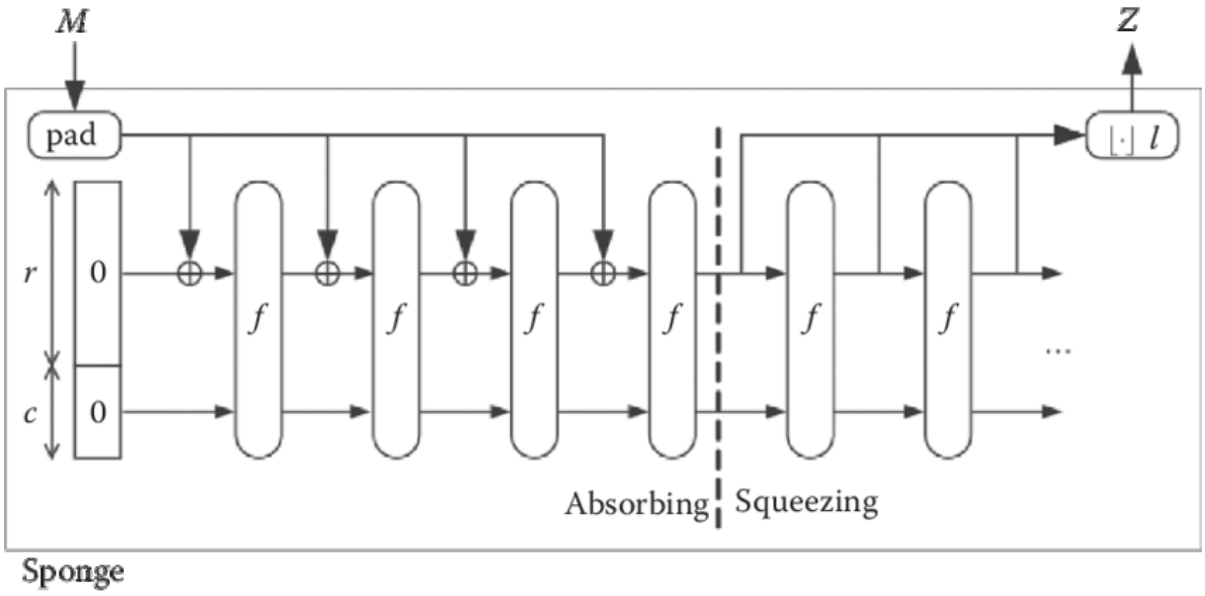


**Figure 8:** Sponge Construction

Source:(https://www.researchgate.net/publication/221274795_Duplexing_the_Sponge_Single-Pass_Authenticated_Encryption_and_Other_Applications/figures?lo=1)

The SHA3-512 is thus in layers, the overall layer KECCAK[c] which results from the sponge construction layer which in turn depends on the KECCAK-p permutation layer.

For SHA3-512; KECCAK[1024] (*N*, 512) = SPONGE[*KECCAK-p*[1600, 24], ***pad10*1***, 576] (***N***,***d***)

### 5.3 Consensus and Time stamping techniques

Time-stamping and consensus schemes are basically built upon the core fundamental mathematical functions above.

### 5.3.1 Time stamping

The idea of trusted time-stamping simply combines the concept of digital signing and hashing. It arose from the need to certify when a document was created or modified, such that it cannot be backdated or forward-dated (Stuart *et al*., 1991). This generally could involve a centralized server (trusted stamping authority) or distributed nodes appending date and time to a hash value and thereafter digitally signing it, with each timestamp containing the preceding timestamp in its hash. This creates a chain of timestamps enclosed in one-way functions (Nakamoto, 2009). For blockchains, this means, a node takes a block of transactions it wants to publish after completing the proof of work, and digitally signs the timestamped hash value, before broadcasting, and it can be easily verified that the contained timestamped hashes are same across the network.

That is recursively;

*Digitally Signed HASH [Digitally Signed (HASH (Transaction) + Timestamp)]*

The time is not an approximate value but is required to satisfy certain conditions give or take a few minutes before the current time. In Bitcoin for example, the timestamp can be accepted if its less than the network time with an addition of two hours, and greater than the median timestamp of the preceding eleven blocks.

### 5.3.2 Consensus schemes

Time-stamping as explained in previous section requires that the nodes of a network reach an agreement on the validity of the time-stamp, likewise block confirmations necessitates the need of such consensus. This is quite important in a decentralized system, due to the lack of trust in such decentralized networks, as Satoshi (2009) pointed out, a system for consensus enables the nodes to agree on a single version of history of transactions. There are quite a few consensus schemes available including those mentioned in the chapter two of this literary work, but in this section the emphasis will be on describing the two most popular schemes.

- **Proof-of-Work schemes (PoW):** - This scheme is essentially used in deciding which nodes have the right to append new blocks to a blockchain and is more of a race of who can solve a puzzle fastest. First, a node obtains the difficulty level from the network which is based on the hash rate of the network, and then gathers all the pending transactions in the network following the last published block and creates a merkle root hash of these transactions. This is combined

with a 32-bit numeric nonce value (Mingxiao *et al*., 2017). The idea is to keep trying different values from 0 to $2^{32}$ as the nonce value in order to obtain a merkle root hash digest which satisfies a given network condition, such as how many zeros the digest must begin with. It is basically a brute force attempt. The first node which is able to achieve this, then has the right to publish the block which other nodes can then verify it satisfies the given conditions. The difficulty level is adjusted every 2,016-blocks which takes roughly two weeks.

For example, the SHA-256 digest value for string *"Justin"* is;

**"db2dc8cff5fe54243815a0252b8950a3769d87ca4bf8d6707ba4ca8b8db9435a00993ef1f4e2f2ac085d82c8a35f9247419923ed135a88baac086b0595f65c85"**.

Subsequently, the concatenation *"Justin1245"* could obtain a digest which starts with one leading zero;

**"0290461C3931C4CF3A3AFBEFD7884DEF3504FC20EA41B0B7E5A85F3F6C86CA32"**

or some other combination. While for the string *"blockchain"* and the nonce value *"934224174",* that is, *"blockchain934224174"* results in a seven leading zero output; **"0000000E2AE7E4240DF80692B7E586EA7A977EACBD031819D0E603257EDB3A81"**.

The difficulty lies in being the fastest to guess a nonce value satisfying that requirement. This is the reason why there are mining farms which break down the nonce values into intervals and share the workload as well as gains.

- **Proof-of-Stake schemes (PoS):** - The PoS scheme in general is based on the hypothesis that the more stake a node has in a system the less likely it would attack the system. Hence, the consensus algorithm utilizes a variety of methods which includes random selection of users with high stakes, calculation of the value of a node's cryptocurrency multiplied by the age of the coin or having the mining difficulty be inversely proportional to the age of the coins a node holds, and other stake-related methods. Thus, a node which has 35% stake for example would be 33% more likely to be selected than one with 2% stake (Yaga *et al.*, 2018).

# 6. ASSESSING BITCOIN AND ETHEREUM CRYPTOGRAPHIC SECURITY

## 6.1 Bitcoin cryptocurrency

In this section, a closer look will be taken at a blockchain-based case study Bitcoin, with an assessment of Bitcoin's cryptographic security in order to identify key considerations which can be generalized to other systems

The Bitcoin cryptocurrency is built on the blockchain technology and is heavily reliant on most of the concepts discussed in the previous chapter. The main components in the Bitcoin system to be discussed in this section includes the hashing algorithm utilized, the digital signature algorithm, the coin selection algorithm and overall flow of transactions. These are the fundamental structural components required.

### 6.1.1 Inner workings and internal structure

Typically, if for example Alice wants to send some Bitcoins to Bob in a transaction, the transaction input is basically a combination of various previous transaction outputs. This method of transaction is called the unspent transaction output (UTXO) scheme in which the number of transactions (Bitcoins) which would equal or exceed the total sum of what Alice wants to send to Bob would be used in generating the transaction, and any balance left would be returned to her.

In simplistic terms, a transaction in Bitcoin cryptocurrency involves a user generating a key pair, which consists of a public key and private key, with the address of that transaction being a representation of the public key (Alqassem and Svetinovic, 2014). Subsequently, the transaction is signed with the sender's private key which can be verified utilizing his known public key. Thereafter, each transaction is confirmed using a proof of work (PoW) consensus scheme and then encapsulated in a block which can contain one or more transactions. Any node or pool of nodes in the Bitcoin network in theory can create a block for any initiated transaction. This is done by solving a cryptographic puzzle, in other words finding a valid proof of work, and this process is called mining.

Typically, the node which succeeds in generating the block would have been the fastest node to create through a random search, a hash value of the block and its contents including previous block hash value, timestamp hash and merkel tree hash of transactions. This hash value has to satisfy the

network difficulty requirements specified by a randomly generated nonce value, such that the generated hash value would be less than or equal to a set network target. The mining node is in turn rewarded with some Bitcoins generated.

The first transaction in a block named the coinbase transaction or generation transaction permits claiming the block reward. It can only be considered spent after a certain number of blocks, due to the phenomenon of forking where multiple valid transaction branches could exist until the network trims it down by accepting the longest chain. Although, there are some wallet implementations which discard transactions that are less than a certain minimum threshold, these transactions are called Bitcoin dust. In other words, the mining fee for instance could be higher than the transaction amounts. While some wallet implementations also drop transactions with excessively high fees. These dropped transactions require being re-initiated (Alqassem and Svetinovic, 2014).

The conditions and criteria required for successful Bitcoin transaction as well as validation routine and hash of ECDSA public key are specified in a stack-based scripting language called script which is intentionally not Turing-complete (Khalilov and Levi, 2018; Bonneau *et al*. 2015).

It is important to note that an attacker in a cryptographic perspective in the Bitcoin context would have to compete against the entire network and would have to back track all previous transactions chained to the particular transaction being attacked.

### 6.1.2 Bitcoin cryptographic algorithms

The most relevant aspects here from a mathematical-cryptographic perspective are the hashing algorithms used, the digital signing algorithm and coin selection algorithm. These aspects are discussed below.

- **Coin selection algorithm:** The dilemma of which coins (transactions) to select for use in a Bitcoin transaction is solved as knapsack problem. The goal of this knapsack algorithm is to select as input the coins which have the highest priority and minimize the number of transactions needed as input (Alqassem and Svetinovic, 2014). Details of the knapsack problem can be found in the work of Smart (2016), and cryptosystems have been built on this concept.
- **Bitcoin hashing algorithm:** The mining process in Bitcoin as mentioned is based on a PoW scheme and this PoW algorithm called Hashcash is built upon the SHA256 hashing

algorithm. The SHA256 is a part of the SHA-2 families discussed in the previous chapter, though originally SHA1 was used in the very first Hashcash version, and then later changed to SHA256. The required property for Hashcash is a partial preimage resistance or one-way functionality (Rudlang, 2017), and it uses two iterations of SHA256 to offer more resistance to hash collisions (the birthday attack), which is essentially SHA256 (SHA256 (Block_Header)).

- **Bitcoin digital signing algorithm:** Typically, the public key generated by a transaction initiator is hashed by Script using an ECDSA on curve secp256k1, and this hashed key is then propagated to the network (Bonneau *et al*. 2015). This ECDSA is based on the Koblitz curve which is a particular type of an elliptic curve, on which point doubling are efficient and curves with large order are easy to find (Bjoernsen Kristian, 2015). These ECDSA signatures are basically used to authorize Bitcoin ownership or transfer and are mathematically generated from the hash of the transaction.

## 6.2 Ethereum cryptocurrency

Ethereum came up as an improvement to the already existing blockchain-based cryptocurrencies. It essentially replaces Bitcoin's "longest chain" consensus rule with a "heaviest subtree" consensus rule. It similarly combines the concept of scripting and hash chaining but rather than a limited stack-based script, utilises a Turing-complete programming language. The aim was to develop an integrated flexible system which can be built upon to serve other functions such as smart contracts. Many transactions have a contract id linking them to one particular contract and due to the Turing-complete language used, transactions can occur in loops. Thus, rules for ownership, and criteria for “unlocking” a contract can be written upon the provided foundation (Wood, 2017).

### 6.2.1 Ethereum accounts

While Bitcoin was built upon the unspent transaction output model, Ethereum is built upon an account-based model. This means an account is the building block and a transaction in this context simply means the transfer of information or “value” between accounts.

There are two general types of accounts which are externally owned accounts and contract accounts. Externally owned accounts are controlled by private keys in a manner is quite similar to the Bitcoin transaction and such that an externally owned account can initiate transfers by creating a transaction and signing it. While contract accounts are governed by a contract code (Vitalik,

2013). When a contract account receives a message, its contract code becomes activated and this makes it possible to store in the internal memory and thus be able to create contracts.

An account basically has a nonce value which indicates the number of transactions issued from that address and is used to ensure transactions are processed only once or indicate number of contracts created by that account, an ether balance which is the basic currency unit of Ethereum, a contract code, and a storage.

### 6.2.2 Inner workings and internal structure

One important concept in Ethereum is the idea of computational limits known as Gas, which is quite relevant with Ethereum being based on a Turing-complete language which is capable of infinitely running loops. The values StartGas and GasPrice in each transaction indicates the limits to number of computational steps which can be executed and the transaction fee for each such step; hence if there is not enough Gas to complete a transaction. The transaction fees for already computed steps are deducted for the miners and the process terminated with an error message (Wood, 2017).

In Ethereum, though currently utilizing a PoW scheme, the use of a proof of stake (PoS) consensus scheme has been proposed and could take effect at some point in the nearest future. This would merge some PoW concepts with the requirements that miners or validators need to have a tangible amount of Ethereum locked as theirs in a master node, the age of these ethers and the amount. While this PoS scheme might favour those nodes who have more, it drastically reduces the power consumption required.

Similarly, from a mathematical and cryptographic point of view, some main areas of concern are the hashing algorithms used and digital signature algorithms. Ethereum also utilizes elliptic curve cryptography for digital signing, while Ethereum’s proof of work function Ethash is built upon the Keccak-256 which is quite similar to SHA-3 hash function described in the previous chapter.

In Ethereum's Keccak, the number of rounds in the hashing algorithm depends on the size of the lane. Randomization is achieved by the sponge construction and Keccak permutation, an iterative permutation process which includes rotating bits in a lane by a constant value, transposing lanes, and applying an S-box to rows.

One of the important factors is that the hash function is deterministic as the signature scheme utilises both a hashing algorithm in addition to a signing algorithm. Hence it is important that the function is injective making each hash distinct and also deterministic always providing the same output as a result of passing through the same state sequence. The hashing algorithm also aims to add collision resistance by using ASCII decimal encoding of the number of bytes in the word being encoded.

### 6.3 Weaknesses and common attacks in cryptocurrencies

Like most cryptography-based systems, Bitcoin and other blockchain-based technologies inherit the weaknesses in their cryptographic functions such as pseudo-collision attacks associated with SHA-2, side channel attacks, timing attacks, Shor's quantum algorithms and computing attacks.

There are a lot of weaknesses in cryptocurrencies, which ranges from vulnerability of wallets, private keys theft or compromise, double spending, to energy consumption issues. In table 1 below, a brief description of some of the most common attacks against cryptocurrencies is provided (Conti *et al*.,2017; Bag *et al*.,2017; Gervais *et al*.,2015; Gervais *et al*.,2016; Apostolaki *et al*.,2017).

**Table 1:** Common attacks against cryptocurrencies

| Cryptocurrency attack | Brief description of attack |
|---|---|
| 51% Attack(50%hash-rate) | An attacker or group controls more than 50% of the network hashrate. |
| Block Withholding attack | An attacker in a mining pool hijacks and submits only partial PoW criteria. |
| Finney Attack | An attacker broadcasts a pre-mined block to use in double spending. |
| Double Spending Attack | An attacker uses the same coin for multiple transactions. |
| Vector-76 Attack | An attacker combines Finney and double spending attacks (multiple). |
| Selfish Mining | Group of attackers create and hide new blocks to create a fork. |
| Wallet/Private Key Theft | An attacker compromises the private key of users or steals their stored wallets. |
| Time Jacking Attack | An attacker speeds up clocks of majority of miners. |
| Sybil Attack | An attacker creates multiple virtual identities. |
| Eclipse Attack | An attacker tampers or delays the sending of transactions/blocks to a node. |
| Deanonymization Attacks | Anonymity issues due to coin history, resulting in linking IP addresses to wallets. |
| Routing Attacks | An attacker isolates a set of nodes delaying/stopping block propagation. |
| DDoS Attack | Hoarding network resources to deny users from services. |
| Long Range Revisions Attacks | Coalition of validators withdrawing their deposits and still using past supermajority to resolve fork/checkpoint conflicts. |
| Slide Attacks | An attacker tries to break down a cryptosytem into some rounds using a function and then analyses key schedules in order to exploit weaknesses. |
| Length Extension Attacks | Attacks which try to add some additional information to a given hash digest and produce valid hashed messages. |
| Timing Attacks | Typical cryptosystem attack where attackers try to analyse time it takes to perform a function or algorithm and thus gain some information. |
| Pre-Image Attacks | When an attacker tries to find a message mapped to a specific hash digest. |
| Pseudo-collision attacks | An attacker focuses on the state-update parts of a hash function and looks for a collision which can lead to breaking the entire hash function. |

## 6.4 Blockchain hash function assessment framework

In the previous sections, there has been an elaboration on the hashing algorithms utilised by both Bitcoin and Ethereum, specifically SHA-2 and KECCAK-256. Most activities in cryptocurrency processing and similarly blockchain usage relies heavily on hashing, this includes and is not limited to digital signing, consensus algorithms, and time stamping. Hence, the aim of this section and the core of this literary work is to analyse the needs of hash functions in blockchain and identify relevant security factors for consideration while taking into account complex requirements for performance.

The conceptual model for assessment proposed in this section, builds upon and combines some concepts from the models established in literature particularly, Damaj and Kasbah's (2018) framework for cryptographic software and hardware implementation; Gervais' *et al*.(2016) framework on security and performance parameter implications in PoW blockchains; and Bauspiess and Damms' (1992) study on functional and quality requirements for hash functions in cryptographic applications. This conceived model focuses on a narrower perspective for blockchain applications.

### 6.4.1 Relevant security factors and complexity attributes

For a hash function to be used in a cryptographic-secure context, in addition to being easy to verify, and producing a fixed length output, it must be a one-way function. In other words, it needs to satisfy the preimage resistance (Smart, 2016). Though these are not the only factors considered when assessing hash function, these security requirements need to be balanced against performance requirements, as it would not be very useful to have a hash function which is quite secure but impractical to use. This is especially important when considering the limitations imposed by size and power in lightweight devices such as RFIDs. In this section therefore, a model which puts these different factors into consideration in assessing hash function to be used in a blockchain context is presented as in Figure 10, and in the order illustrated in Figure 9.

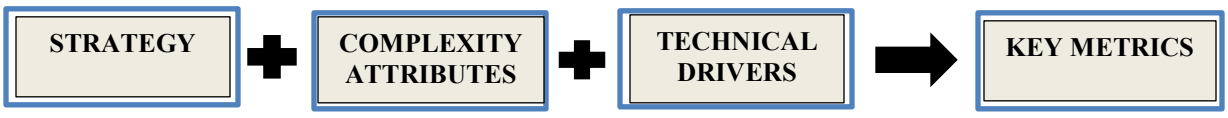


**Figure 9:** Factors to consider in Assessment

First, as with any technology, choices and assessments are governed by and impacted upon by the strategic goals which is intended for that technology. Similarly, the factors which makes the choice of a hash function or its assessment more complicated and difficult, such as size limitations and power requirements are all considered and presented as complexity attributes. Lastly, the basic essential technical requirements and drivers are considered bearing in mind the strategy and complexities. Armed with these, suitable metrics and choices which will be most appropriate and fit for purpose can be made.

### 6.4.2 Proposed assessment model

This proposed assessment model is presented as high-level conceptual blocks of factors, under the four crucial categories mentioned above as shown in Figure 9. Typically, this applies to at least Ethereum and Bitcoin implementations but can be generalized to other blockchain-based applications and other applications which utilize cryptographic hash functions.

PoW-based blockchains Bitcoin and Ethereum considered factors such as partial preimage resistance and collision resistance, which led to Bitcoin's use of two iterations of SHA256. This is also related to the use of randomization techniques such as Keccak permutations. Similarly, due to the use of their hashing algorithm in their digital signing schemes, it was necessary to have determinism and injectivity in the hashing algorithms.

While Ethereum is more difficult to implement with ASIC (application specific integrated chips) chips with an increased memory requirement, Bitcoin with increasingly high mining hash power requirements favours the use of special ASIC chips and subsequently higher hash rates and promotes random search strategies. The dependency on special chips also impacts the choice of a hashing algorithm, as any change in a hashing algorithm for a blockchain could render the ASIC in use obsolete.

From a strategical point of view, the goal of the hash process will determine to a huge extent which factors can be overlooked in order to maintain a sensible trade-off between efficiency and security. In the same vein, the context of usage has a great weight on which criteria is impossible to ignore and determines the level of security required.

The main factors identified as complexity attributes in this model which have an impact on decisions regarding security requirements are as follows;

- the hash rate which is directly linked to the amount of power being consumed continuously,
- the memory storage requirement (size of digest and input) and their associated ASIC Cost (gate equivalents),
- known weaknesses of core components and relevant known attacks,
- computational power boundaries,
- the ease of implementation and ease of obtaining the algorithm,
- compatibility with differing machines, and
- the hash calculation speed.

Then, this is combined with the typical technical requirements of a cryptographic hash function which includes the following;

- the hash function having collision resistance, preimage collision resistance and second preimage collision resistance,
- the number of rounds used (a huge factor in pre-image attacks),
- the usage of randomization elements,
- having a fixed output size,
- being able to handle input size of any length,
- the absence of decomposition,
- having evenly distributed preimages in output,
- compression efficiency, and
- the probability of collisions.

Subsequently, the key metrics presented are those criteria which regardless of the context should be satisfied by any hash function before it can be called a cryptographic hash function. Now these, depending on the usage, context and strategic needs, other factors can be added upon these. These includes but are not restricted to the absence of secret seeds which can result to slide attacks, having an efficient verification process, being preimage collision resistant at the least, being difficult to decompose into smaller functional units, having a fixed output size, having a low probability of collisions occurring, and having evenly distributed preimages in output.

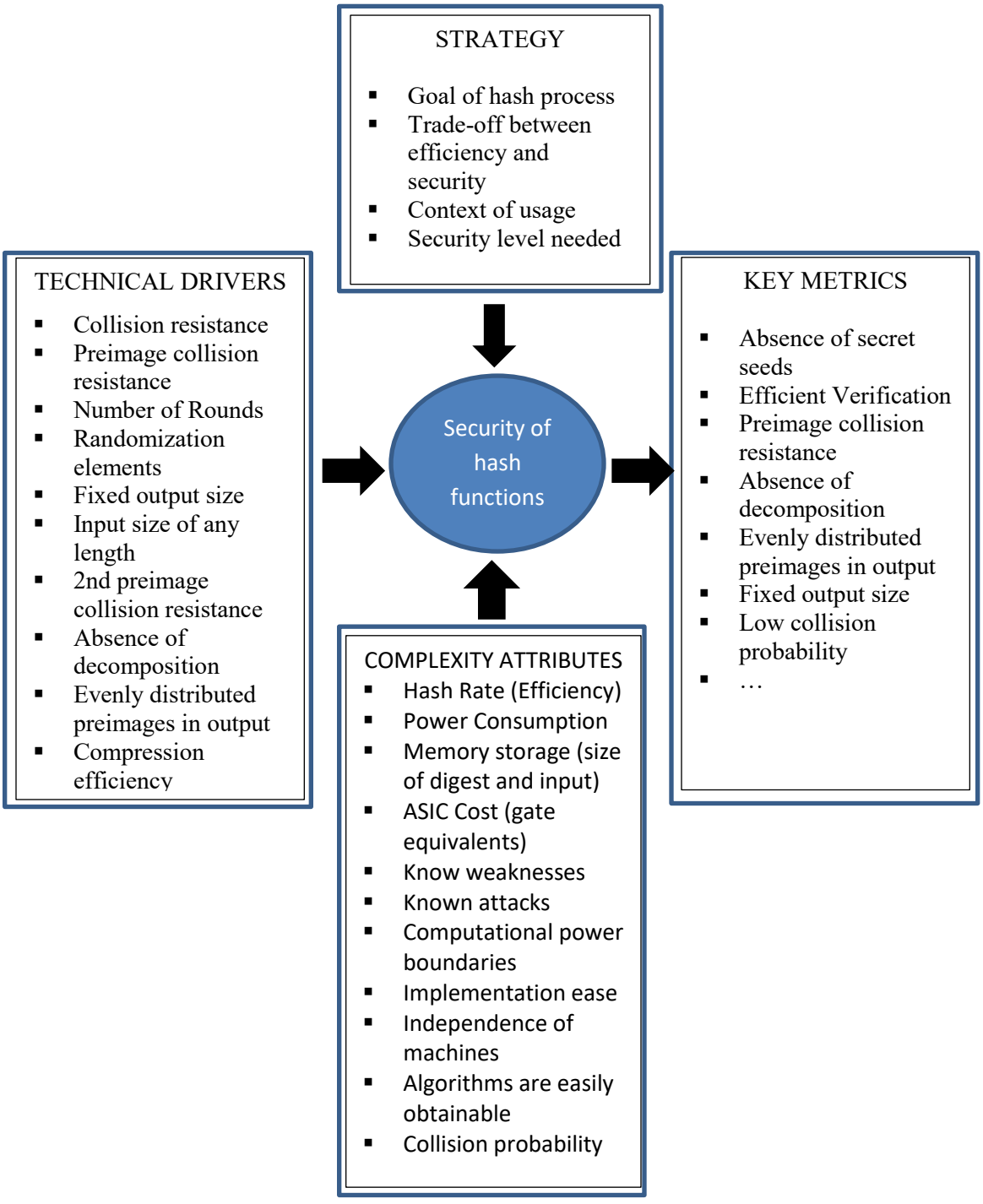


**Figure 10:** Hash Function Security Assessment Model

# 7. CONCLUSIONS

Blockchain-based applications are rapidly having huge impacts across various industries most especially the financial sector, and changing drastically how documents, contracts and information are handled in a lot of processes. These changes are expected to keep increasing as businesses realize the potential benefits associated with such blockchain-based systems. In the same vein, it is becoming increasing more important to focus a lot of research attention towards gaining a deeper understanding on how to make this transition better and help improve the systems in question.

The security of blockchain technology covers a broad range of topics which can fall into many different categories such as physical security, network security, internet security, cloud security, application security, data security and cryptographic security. This thesis focused on the underlying concepts which affects the cryptographic security of blockchain technology. These are mainly the hash functions used in transforming the data involved in transactions and making them extremely difficult to change, as well the digital signature schemes utilised in ensuring privacy. This focus on cryptographic security essentially cut across many areas simultaneously, particularly data security and network security by discussing consensus schemes and hashing algorithms.

This study has been focused on making this process of understanding the technology a lot easier from a cryptographic perspective for students studying cryptography. To provide a good foundation and starting point, a brief introduction to the mathematical notations and concepts on which blockchain is built upon was introduced, as well as an overview of the technology and its operations. Special attention was given to the two identified crucial blockchain building blocks; hash functions and digital signatures, with emphasis on the mathematics behind them. Based on the core blockchain principles, concepts, ideas and previous research, a model for assessing hash functions was presented. The outcome of this research helps provide guidance and direction in which factors one should consider when selecting hash functions to be used in a blockchain-based system.

The factors proposed in the generated model can be categorized into three main areas which are strategical factors, complexity attributes and technical drivers. When these categories of factors are combined in a particular context of usage, suitable metrics can then be extracted for assessing the hash function used. This research agrees with and build upon the works of Damaj and Kasbah's

(2018), Gervais' *et al*.(2016), and Bauspiess and Damms' (1992) regarding frameworks for hash functions. The model provided is not constrained to blockchain applications, as it has relevance to any application and technologies which rely on hash functions. The most relevant factor is the context in which such assessment is being made, and thus how extreme the security restrictions should be.

There is room for improvement and further research on smart contracts in particular, as this is one major application in blockchain technology which is expected to have the most extensive impact in future. Currently, research is still ongoing as to implementation of a new proof of stake scheme Casper in Ethereum, and there are quite many limitations in the scaling of these blockchain-based systems, and this provides an opportunity for more research.

From a cryptographic security perspective, the two most relevant components of a blockchain are the implementations of hashing and digital signing, and the factors governing these implementations are dependent on the context. At the bare minimum, any cryptographic hash function used in a blockchain would be better equipped to deal with threats from a security perspective when there is an absence of secret seeds, uneven preimage distribution and varying output sizes, as well as manifestation of efficient verification of hashes, preimage collision resistance and a low collision probability. These provide a solid base for hashing applications.

# REFERENCES

Ali Kaan Koç – Emre Yavuz – Umut Can Çabuk – Gökhan Dalkiliç (2018) Towards Secure E-Voting Using Ethereum Blockchain. *Conference: International Symposium on Digital Forensic and Security (ISDFS), Antalya,* Vol. 6

Alqassem, Israa – Svetinovic, Davor (2014) Towards Reference Architecture for Cryptocurrencies: Bitcoin Architectural Analysis. *IEEE International Conference on Internet of Things (iThings 2014), Green Computing and Communications (GreenCom2014), and Cyber-Physical-Social Computing (CPSCom 2014),* 436 – 443.

Anceaume, Emmanuelle – Lajoie-Mazenc, Thibaut – Ludinard, Romaric – Sericola, Bruno (2016) Safety Analysis of Bitcoin Improvement Proposals. *IEEE 15th International Symposium on Network Computing and Applications,* 318-325. 10.1109/NCA.2016.7778636.

Andersen, P. – Kragh, H. (2010) "Sense and sensibility: two approaches for using existing theory in theory building qualitative research", *Industrial Marketing Management*, Vol. 39, 49 – 55.

Apostolaki, Maria – Zohar, Aviv – Vanbever, Laurent (2017) Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. *Conference: IEEE Symposium on Security and Privacy (SP – 2017),* 375–392. 10.1109/SP.2017.29.

Bag, Samiran – Ruj, Sushmita – Kouichi, Sakurai (2017) Bitcoin Block Withholding Attack: Analysis and Mitigation. IEEE Transactions on Information Forensics And Security, Vol. 12, No. 8, 1967 – 1978

Banerjee, Mandrita – Lee, Junghee – Choo, Kim-Kwang Raymond (2017) A Blockchain Future to Internet of Things Security: A Position Paper. *Digital Communications and Networks*.

Bauspiess, Fritz – Damm, Frank (1992) Requirements for Cryptographic Hash Functions. *Computers and Security* 11*,* 427 – 437.

Bjoernsen, Kristian (2015) Koblitz Curves and its practical uses in Bitcoin security. Available: https://koclab.cs.ucsb.edu/teaching/ecc/project/2015Projects/Bjoernsen.pdf [Accessed May 31, 2018]

Böhme, Rainer – Christin, Nicolas – Edelman, Benjamin – Moore, Tyler (2015) Bitcoin: Economics, Technology, and Governance. Journal of Economic Perspectives, Vol. 29, No.2, 213-238.

Bonneau, Joseph – Miller, Andrew – Clark, Jeremy – Narayanan, Arvind – Kroll, Joshua A. – Felten, Edward W. (2015) SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. *IEEE Symposium on Security and Privacy,* 104 – 121.

Buchmann, Johannes – Lindner, Richard – Rückert, Markus – Schneider, Michael (2009) Post-quantum cryptography: lattice signatures. *Computing,* Vol. 85, Issue 1-2, 105 – 125.

Chi, Lianhua – Zhu, Xingquan (2017) Hashing Techniques: A Survey and Taxonomy. *ACM Computing Surveys,* Vol. 50, No. 1, 1 – 36. 10.1145/3047307.

Conti, Mauro – Kumar, Sandeep E. – Lal, Chhagan – Ruj, Sushmita (2017) A Survey on Security and Privacy Issues of Bitcoin, Retrieved April 14, 2018, from https://arxiv.org/pdf/1706.00916.pdf

Creswell, J. W. (2009). *Research design: Qualitative, quantitative, and mixed methods approaches* (3rd ed.). Thousand Oaks, CA: Sage

Dai, Fangfang – Shi, Yue – Meng, Nan – Wei, Liang – Ye, Zhiguo (2017) From Bitcoin to Cybersecurity: a Comparative Study of Blockchain Application and Security Issues. 4th International Conference on Systems and Informatics (ICSAI - 2017), 975 – 979.

Damaj, Issam – Kasbah, Safaa (2018) An Analysis Framework for Hardware and Software Implementations with Applications from Cryptography. *Elsevier: Computers and Electrical Engineering* 69*,* 572 – 584.

De Meijer, Carlo R. (2015) The UK and Blockchain technology: A balanced approach, Journal of Payments Strategy & Systems, Vol. 9, No. 4, 220-229.

Dinh, Tien T. A. – Liu, Rui – Zhang, Meihui – Chen, Gang – Ooi, Beng C. – Wang, Ji (2017) Untangling Blockchain: A Data Processing View of Blockchain Systems. *Cornell University Computing Research Repository CoRR, abs/1708.05665.*

Dobraunig, Christoph – Eichlseder, Maria – Mendel, Florian (2015) Analysis of SHA-512/224 and SHA-512/256. *Proceedings, Part II, of the 21st International Conference on Advances in Cryptology - ASIACRYPT 2015,* Vol. 9453, Pages 612-630

Dresh, Aline – Lacerda, Daniel P. – Miguel, Paulo A.C. (2015) A Distinctive Analysis of Case Study, Action Research and Design Science Research. *Review of Business Management,* Vol. 17, No. 56, 1116 – 1133.

Gervais, Arthur – Karame, Ghassan O. – Wüst, Karl – Glykantzis, Vasileios – Ritzdorf, Hubert – Capkun, Srdjan (2016) On the Security and Performance of Proof of Work Blockchains. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS - 2016)*, 3 – 16.

Gervais, Arthur – Ritzdorf, Hubert – Karame, Ghassan O. – Capkun, Srdjan (2015) Tampering with the Delivery of Blocks and Transactions in Bitcoin. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS – 2015)*, 692 – 705.

Haber, Stuart – Stornetta, Scott (1991) How to time-stamp a digital document, Journal of Cryptology, Vol 3, No 2, 99-111.

Hanifatunnisa, Rifa – Rahardjo, Budi (2017) Blockchain-based E-Voting Recording System Design. *2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA)*, 1-6.

Howe, James – Pöppelmann, Thomas – O'Neill, Maire – O'Sullivan, Elizabeth – Guneysu, Tim (2015) Practical Lattice-Based Digital Signature Schemes. *ACM Transactions on Embedded Computing Systems*, Vol. 14, No. 3, 41: 1-24.

Hudic, Aleksandar – Smith, Paul – Weippl, Edgar R. (2017) Security Assurance Assessment Methodology for Hybrid Clouds. *Computers and Security,* Vol. 70, 723 – 743

Kaushal, Puneet Kumar – Bagga, Amandeep – Sobti, Rajeev (2017) Evolution of Bitcoin and Security Risk in Bitcoin Wallets. *International Conference on Computer, Communications and Electronics (Comptelix - 2017),* 172 – 177.

Khalilov, Merve C. K. and Levi, Albert (2018) A Survey on Anonymity and Privacy in Bitcoin-like Digital Cash Systems. *IEEE Communications Surveys & Tutorials* 12 (4).

Korpela, Kari – Hallikas, Jukka – Dahlberg, Tomi (2017) Digital Supply Chain Transformation toward Blockchain Integration. *Proceedings of the 50th Hawaii International Conference on System Sciences (HICSS 2017),* Vol. 50, 4182 - 4199.

Letourneau, K. B – Whelan S. T. (2017) Blockchain: Staying Ahead of Tomorrow. *Journal of Equipment Lease Financing*, Vol. 35, No. 2, 1 - 7.

Li, Xiaoqi – Jiang, Peng – Chen, Ting – Luo, Xiapu – Wen, Qiaoyan (2018) A survey on the security of blockchain systems. *Elsevier - Future Generation Computer Systems*, abs/1802.06993.

Liu, Yi – Li, Ruilin – Liu, Xingtong – Wang, Jian – Zhang, Lei – Tang, Chaojing – Kang, Hongyan (2017) An Efficient Method to Enhance Bitcoin Wallet Security. *11th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID - 2017),* 26 – 29, ICASID.2017.8285737

Luo, Jingwei – Huang, Alex Q. (2017) Bit-Energy: An Innovative Bitcoin-style Distributed Transactional Model for a Competitive Electricity Market. *Conference: 2017 IEEE Power & Energy Society General Meeting,* 1-5. 10.1109/PESGM.2017.8274639.

Mehta, Inderpal Singh – Chakraborty, Arnav – Choudhury, Tanupriya – Sharma, Mukul (2017) Efficient Approach towards Bitcoin Security Algorithm. *International Conference on Infocom Technologies and Unmanned Systems (ICTUS - 2017),* 807-810. 10.1109/ICTUS.2017.8286117.

Menezes, Alfred – Van Oorschot, Paul – Vanstone, Scott (1996) *Handbook of Applied Cryptography*. CRC Press, Inc. Boca Raton, FL, USA.

Mingxiao, Du – Xiaofeng, Ma – Zhe, Zhang – Xiangwei, Wang – Qijun, Chen (2017) A Review on Consensus Algorithm of Blockchain. *IEEE International Conference on Systems, Man, and Cybernetics (SMC - 2017)*, Banff, Canada, 2566 – 2572.

Nakamoto, Satoshi. (2009). Bitcoin: A Peer-to-Peer Electronic Cash System. Available: https://Bitcoin.org/Bitcoin.pdf [Accessed June 6, 2018]

National Institute of Standards and Technology (2015), *FIPS PUB 202 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: SHA-3 Standard: Permutation,* Gaithersburg, MD: Information Technology Laboratory.

National Institute of Standards and Technology (2015), *FIPS PUB 180-4 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION: Secure Hash Standard (SHS),* Gaithersburg, MD: Information Technology Laboratory.

Natoli, Christopher – Gramoli, Vincent (2017) The Balance Attack or Why Forkable Blockchains Are Ill-Suited for Consortium. *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks,* 579 – 590.

Neisse, Ricardo – Steri, Gary – Nai-Fovino, Igor (2017) A Blockchain-based Approach for Data Accountability and Provenance Tracking. *Proceedings of the 12th International Conference on Availability, Reliability and Security - ARES '17,* 1 – 10.

Paar, Christof – Pelzl, Jan (2010). *Understanding Cryptography: A Textbook for Students and Practitioners.* Springer, Verlag Berlin Heidelberg, New York.

Park, Keon Chul – Shin, Dong-Hee (2017) Security Assessment Framework for IoT Service. *Telecommunications Systems,* Vol 64, Issue 1, 193 - 209.

Pontiveros, Beltran B. F. – Norvill, Robert – State, Radu (2018) Recycling Smart Contracts: Compression of the Ethereum Blockchain. *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS - 2018)*, 1-5. 10.1109/NTMS.2018.8328742.

Pustišeka, Matevž – Kos, Andrej (2017) Approaches to Front-End IoT Application Development for the Ethereum Blockchain. *Elsevier - International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI2017), Procedia Computer Science 129*, 410 – 419

Rudlang, Marit (2017) Comparative Analysis of Bitcoin and Ethereum. *Thesis Work*, *Norwegian University of Science and Technology,* 1 – 103.

Sato, Tatsuya – Himura, Yosuke (2018) Smart-Contract based System Operations for Permissioned Blockchain. *9th IFIP International Conference on New Technologies, Mobility and Security (NTMS - 2018)*, 1-6.

Sharma, Pradip Kumar – Park, Jong Hyuk (2018) Blockchain-based hybrid network architecture for the smart city. *Elsevier - Future Generation Computer Systems*, 10.1016/j.future.2018.04.060.

Smart, Nigel P. (2016). *Understanding Cryptography: A Textbook for Students and Practitioners.* International Publishing, Switzerland.

Stefan, Wolf (1997) Entropy Measures and Unconditional Security in Cryptography (Doctoral dissertation, Swiss Federal Institute of Technology, Zurich). *Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.1016&rep=rep1&type=pdf*

Stewart, Jenny (2012) Multiple-Case Study Methods in Governance-related Research. *Public Management Review,* Vol. 14, Issue 1, 67–82.

Vitalik Buterin (2013) Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. *Retrieved from* http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentr alized_application_platform-vitalik-buterin.pdf.

Vranken, Harald (2017) Sustainability of Bitcoin and blockchains. *Current Opinion in Environmental Sustainability 2017*, 28: 1–9.

Vujičić, Dejan – Jagodić, Dijana – Ranđić, Siniša (2018) Blockchain Technology, Bitcoin, and Ethereum: A Brief Overview. *17th International Symposium INFOTEH-JAHORINA,* 1–6. 10.1109/INFOTEH.2018. 8345547.

Wood, Gavin (2017) Ethereum: A Secure Decentralized Generalized Transaction Ledger. *Ethereum Yellow Paper.*

Woodside, Joseph. M – Augustine, Fred K. – Giberson, Will (2017) Blockchain Technology Adoption Status and Strategies, *Journal of International Technology and Information Management*, Vol. 26, No. 2, 65-92.

Yaga, Dylan – Roby, Nick – Scarfone, Karen (2018) Blockchain Technology Overview. *National Institute of Standards and Technology Internal Report 8202*, 1 – 59.

Yanga, Changsong – Chena, Xiaofeng – Xiang, Yang (2018) Blockchain-based publicly verifiable data deletion scheme for cloud storage. *Elsevier - Journal of Network and Computer Applications* 103, 185 – 193.

Zheng, Zibin – Xie, Shaoan – Dai, Hongning – Chen, Xiangping – Wang, Huaimin (2017) An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. *2017 IEEE International Congress on Big Data (BigData Congress).* 557-564.

Zhou, Jianying – Bao, Feng – Deng Robert (2006) Minimizing TTP's Involvement in Signature Validation. *International Journal of Information Security* 5 (1)*,* 37 – 47.