Redactable Blockchain and Chameleon Hash Function Characterization

by

Shruti Vaghani
201811072

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

June, 2020

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

Shruti Vaghani

Certificate

This is to certify that the thesis work entitled Redactable Blockchain and Chameleon Hash Function Characterization has been carried out by Shruti Vaghani for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my/our supervision.

Dr. Maniklal Das
Thesis Supervisor

Acknowledgments

First of all, I would like to thank God for giving me the strength, and good health needed to complete this thesis. I also want to pay my gratitude towards my parents and my friends, for their constant support and faith in me. Without their encouragement and belief, it would have not be complished.

I wish to thank my thesis supervisor Prof. Maniklal Das for his constant guidance and patience over the duration of the thesis work. He constantly motivated me that I will be able to complete the thesis work when I had lost hopes. Thank you for encouraging me and pushing me to the edge to get the best out of me.

I also want to thank the faculties of DA-IICT who have taught me courses during my MTech. Their teaching provided good knowledge, which was helpful during the thesis work.

Finally, I would like to thank every person in the Institute who touched my life during this period of 2 years and helped me learn something new.

Contents

Abstract

Redactable Blockchain – a framework that helps to rewriting or compress the data of blocks in blockchain technology. There are several reasons to use Redactable Blockchain, a necessity to edit or remove the relevant content and the probability to required rewrite data to "the right to be forgotten[3]." Blocks or transaction has been resisted in blockchain; it's persistent and not modifiable. This essential feature of blockchain is desirable -to allow for breaking immutability in a controlled way. This problem has an elegant solution at block level to replace standard hash function to chameleon hash function. The chameleon hash function is a collision-resistant hash function without knowledge of trapdoor (secret information), in regular cryptographic hash function hard to find a collision. However, anyone who knows the trapdoor key efficiently generates pre-image; without trapdoor key, nobody can compute collision even if we can see collision for an arbitrary hash function capacity is distributed among several distributed parties (Bitcoin)[10]. The standard chameleon hash characterized based on non-standard properties in factoring and discrete logarithm. Implementing the optimal solution of the chameleon hash function based on simple factorization, discrete logarithm and advanced factorization to calculate the hash generation time and collision finding time. Comparison study of these solutions needs to be done yet.

Keywords: *Redactable Blockchain, Chameleon Hash Function.*

List of Tables

List of Figures

CHAPTER 1
Introduction

Blockchain is a public ledger grown around a pair to pair system that can be publicly shared among different users to create an unchangeable record of transactions, per timestamped and linked to the previous one. The set of transactions is add in the block of the chain. Blockchain is a system of storing information that makes it difficult or impossible to change, hack, or cheat the system.

The blockchain carries two important aspects: idea of decentralization and complex smart contracts. In a decentralized system, one single entity can not store the information. Everyone in the network owns the data.and the smart contracts are programs to perform simple functions. Once content or block publish in a chain then that cannot modify or erase the data in blockchain technology. The blockchain has three important properties is Decentralization, Transparency, and Immutability. Decentralization means one entity can not control the whole system, and transparency mean all the data is accessible by all the participants, and the immutable means whatever data is added in the database will never change or erase.

In an imperfect world, there are faces many problems with an immutable blockchain. This can be solve using modifies existing blockchain technology to allow designated authorities to edit, rewrite or remove previous blocks of information without breaking the chain, this modified blockchain called redactable blockchain. The blockchain technology contains three essential features which are more useful for business and real-world application. These features are decentralization of data, reducing cost, and easy to trace the transaction done by people.

The blockchain editing can achieve by using a new variation of the "chameleon hash function", which can recreate matching algorithms through the use of trapdoor key. After a change has been made to a block, the original blockchain remains fully intact and there is no need to create a hard fork and rebuild subsequent blocks.

The chameleon hash function is collision resistance hash function unless trapdoor is not known. The knowledge of trapdoor allows finding arbitrary collision in the domain of the function, finding collision time based in their implementation. The chameleon hash function is classified in non-standard property of collision-resistant for signer(sender) and collision tractable for receiver. The hash function and Chameleon hash function are a bit different as hash function is a one-way function, that is not reversible but chameleon hash function is reversible using trapdoor key.

1.1 Motivation

Blockchain, the transaction is Stored in the ledger forever and is immutable. These fit perfectly in the current System, but these immutable ledgers are not appropriate for all applications. Blockchain use to store the data or code. Blockchain can redact its content in some specific and exceptional circumstances. Redactability can be archived in the blockchain using a particular type of hash function (Chameleon hash Function) for Block generation.

Chameleon Hash Function has a trapdoor key. Using this key, we can redact one, or many blocks depends on the design. Nowadays, we believe it is not much effort is spent to promote the "right to forgotten[3]." Blockchain application store the files, notarized documents, manage health document, administrative asset, etc.. The record should me destroy if they contain an error of sensitive information; to find, edit, or erase the confidential information take some time. Which is depends on the implementation of the chameleon hash function. We can modify the construction of the chameleon hash function so that collision finding the time and new hash generation time takes less time.

1.2 Problem Statement

The blockchain can achieve redactibility using a chameleon hash function instead of standard cryptographic hash function[10]. The standard chameleon hash has function four tuples: key generation, hash generation, hash verification, and collision finding[3]. The chameleon hash function will take time to generate a hash and also take time in collision finding. The chameleon hash function has real-time applications like, a sanitizable signature,which is used when an authorized third party wants to modify the signature on the message. It takes some time, which is based on the implementation of a chameleon hash function.

The chameleon hash function can be characterized based on their non-standard properties in simple factorization, discrete logarithm, and advanced factorization. They take some time to generate the hash and to find collision time based on their implementation. We try to implement the different chameleon scheme and to compare their hash generation time and collision finding time.

1.3 Thesis Outline and Contribution

Introduction to the thesis is explained in chapter 1. To better understand the basic concepts related to blockchain, we have done background study related to that, which is there in chapter 2. Basics related to redactable blockchain and chameleon hash function are explained in chapter 3, with classification of their standard properties. My contribution to the thesis is that, I have found out optimal solution for simple factorization and discrete logarithm based on the non-standard properties of the chameleon hash function, in addition to that, solution of advanced factorization is also there. Implementation of above three functions has been done to find collision time and hash generation time. Moreover, i have checked these three follows properties of standard cryptographic hash function or not? As a result of that, I have come up with the result that, collision resistance and efficiency of algorithm has been followed by above written functions. All these things are listed in chapter 4. I have concluded the thesis in chapter 5 with some possible improvements which can be done to the system for better performance.

1.4 Applications of Chameleon Hash Function

- **Sanitizable signature:**
  The scheme allows to the signer that they partially commit signing privileges on a message to another party, called a sanitizer[21]. When a message is signed, the sanitizer changes pre-decided elements of the message and create a new signature for a sanitized message without communicating with the signer. Let's take an example: In a medical report, there is some portion

of data which remains confidential for the patient. But the higher authority can access that data. The digital signature can use it for achieving integrity and authentication.

But there could be circumstances where, any authorized third party can change the data in the document. Sometimes it happens that the original document signer is not available at particular time, and their key is expired. Then, the third party has the power to sign the document on behalf of the original signer of the document. This can be possible using the sanitizable signature.

- **Wireless Sensor Networks (WSN):**
  The wireless sensor network uses sensors to collect the data about weather, temperature, etc., at different locations; they process the data and send it to the destination. Normally, the multihop network has n number of nodes. Each and every node carry one or many sensors. One sensor node transfers data to another sensor node through the base station. The data can not be changed during transmission. The nodes n1 and n2 can communicate with each other after the authentication. The authentication can be achieved using the chameleon hash function, where both node n1 and n2 will share the key n in a secure channel. Now, only the keyholders are able to calculate the hash collision[10].

- **Chameleon Signatures:**
  In this scheme, the signer uses the chameleon hash function to calculate the hash h for the specific message m. These hash h was signed using the signing algorithm(the signing algorithm takes input as a message m and return results as a sign the message using their private key, and it's denoted as 'SIG').

  For example, Alice wants to send a message to Bob, so that no one can read the message, the Alice use the chameleon hash function of Bob, computes the hash h and sign it using a digital signature algorithm (i.e., RSA). When Bob receives the signature, he will able to check that signatures that are they valid or no? For that he needs to compute the hash collision. Here, the signature is nontransferable and also guarantees the authentication.

CHAPTER 2

Background study

In this chapter, we will briefly explain the blockchain and their properties, how it is useful in real-time applications, and which type of attacks are possible in blockchain networks.

2.1 Blockchain

Blockchain is a public ledger grown around a pair to pair system that can be publicly shared among different users to create an unchangeable record of transactions, per timestamped and linked to the previous one[2]. The set of transactions is add in the block of the chain. Blockchain can be updated by consensus between participants in the system, and once new data was inserted, it can never erase. It is a write-once, append-many technology, making it an accurate and auditable record of every transaction. Blockchain is a system of storing information that makes it difficult or impossible to change, hack, or cheat the system.

Blockchain Blockchain provides a peer to peer network[13]. This characteristic of blockchain enables the transactions to connect only two parties, which are the sender and receiver. Hence it eliminates the requirement of 'third party authorization' because all in the network is themselves capable of authorizing the transactions.

The blockchain is primarily a public digital ledger of transactions duplicated and distributed across the entire network of computer systems on the blockchain. Each block in the chain contains many transactions, each and every time a new transaction happens on the blockchain, that transaction record is added to every participant's ledger. The decentralized database maintained by various participants is known as Distributed Ledger Technology (DLT). Such a distributed ledger properties are:

1. **Decentralization**
   Before the cryptocurrency Bitcoin and BitTorrent came along, we were significantly used to centralized services. The idea is elementary. You have one centralized entity, it can store all the data, and you'd have to interact individually with this entity, and you will get the information whatever you required. Let's take an example of a centralized system, the banks save all your money, and when you want to pay someone, it is possible through the centralized entity(bank).

   The centralized systems have fed us well for many years. However, they have specific vulnerabilities.

   - If the system is centralized, then all the data was a store in one place. That makes it easy to target localities for potential hackers.
   - If the centralized system is gone for a software upgrade, it can halt the entire system.
   - If the centralized entity anyhow shuts down for some reason. Then nobody will work capable of accessing the information that it possesses.
   - Worst case scenario, if this entity gets damaged and malicious. Then all the transactions that are inside the blockchain will be endangered.

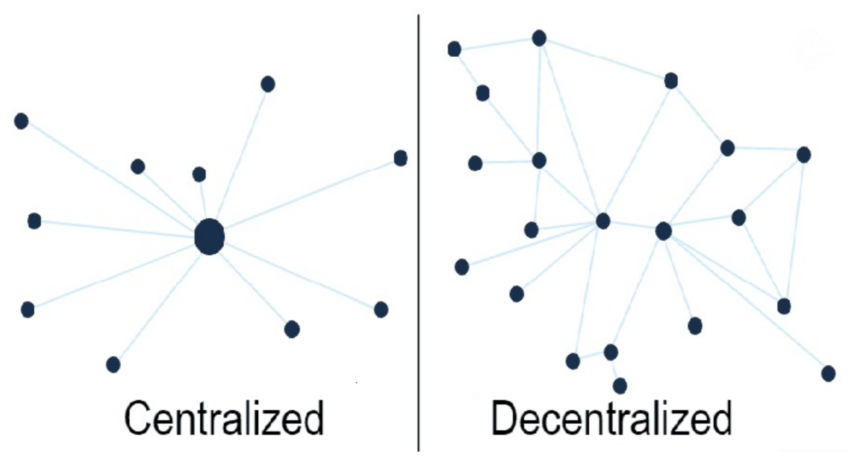

Figure 2.1: The centralized and Decentralized network[20]

In a decentralized system, one single entity can not store the information. Everyone in the network owns the data.If you wanted to communicate with your friend in a decentralized system, you could do it directly without any third party. That was the main philosophy behind Bitcoins. Only you have a charge of your money. You can transfer your money to anyone that you want without going to a bank.

2. **Immutability**
   Immutability relates to the strength of blockchains to prevent modification of transactions, which are already being confirmed. Even though these transactions are usually relating to transfer the cryptocurrencies, they may also commit to the record of non-monetary forms of digital data.

3. **Transparency**

   One of the numerous exciting concepts in the blockchain is "transparency." Some people believe that blockchain provides you privacy, while some people say that it is transparent. A person's identity is hidden through complex cryptography and represented only by their public address. When you look up a person's transaction history, you will not be able to see that **"Bob sent 1 BTC"** , but in the blockchain, you will see **"1MF1bhsFLkBzzz9vpFYEmvwT2-TbyCt7NZJ sent 1 BTC"**[16]. So, that the person's real identity is secure, you will see all the transactions done on their public address. This level of transparency has nevermore been before in a financial system. It adds that extra level of accountability required by some most prominent institutions.

   In a cryptocurrency, you know the public address of one of these big companies. You can directly pop it in an explorer and look at all the transactions that they have engaged; this will force them to be honest. Something that they had never to deal with before. However, that is not the best use-case.

   The most of the companies will not transact using cryptocurrency, and even if they do, they will not do ALL their transactions using cryptocurrency. so that, the person's real identity is secure, you will see all the transactions done on their public address. This level of transparency has nevermore been before in a financial system. It adds that extra level of accountability required by some most prominent institutions.

Txn Hash	Block	Age	From	To	Value	Txn Fee
0xdfac4766b1766951d086fbe71b5094884fa0eca9f6f8c6e7eb47c37d7a3c51e3	10509602	27 secs ago	0x3cd90144617f46876ebf280e99e05b28827872bd	Tether USD	0 Ether	0.00196567
0xcf26ee72a076f1624bca3fb3011eaf2087aa8037b683a7a64dee5af96234ea1b	10509602	27 secs ago	0x616ea3c6009fee1b2b8c9179d157284b5bf759ca	KuCoin	20.02554797169 6925 Ether	0.001575
0xac92f18e1f8b83b9b90658f2b9f48e4e00c27ea9e2b2eaee6473d4040bc9ca89	10509602	27 secs ago	0x42c3cd223fb670bdb9f7414df4982463b353e823	Binance	516.994536326 Ether	0.001575
0x6a85d11467be49e86115dc2042aef94b8b08e0b3769a5c29d0a750087188377d	10509602	27 secs ago	Acronis	Acronis & Acronis: Contract	0 Ether	0.0049344
0x432e2d0e32fda12619b333de370bc5515b38d58feb2d4c8f0fb74fd52fa40b88	10509602	27 secs ago	0x58a71ffc1918bb7cc6ab4a9c4b04baafdedbc701	Binance	20.003043263 Ether	0.001575
0x25465816bc91c8e0413aae752eeb0915714a90a2742ee108d8362555101ea189	10509602	27 secs ago	0x25465816bc91c8e0413aae752eeb0915714a90a2	Binance	2.87686709 Ether	0.001575

Table 2.1: Transaction transparency in blockchain

4. **Secure**
   Blockchains achieve their data security. Through cryptographic hashing functions are of essential importance. Hashing is a process where the algorithm (hash function like SHA-256) receives input as a data of whatever size and returns an output, which is a predictable and fixed length. Hence, the hash of each block is produced based on the data carried within that block and the hash of the previous block. These hash identifiers perform a major role in assuring blockchain safety and immutability.

5. **Anonymous**
   The user's identity is either anonymous or pseudonymous, meaning when people transact bitcoin, from their wallet address, and transaction details are stored in the blockchain. While there is no connection between their wallet address and their identity, these transaction stays anonymous in the blockchain network.

6. **Time-Stamped**
   In a decentralized network, no one can control the database timestamps. When transactions proved that it is valid, then that transaction is stored in the system with a timestamp[15]. All transaction and their timestamp publicly visible.

Data recorded in blockchain is immutable and cannot be modified as described above. Also, the data is stored to the block after to give approval by all participants in the network; they allow secure transactions. These participants who validated the transactions[7] and stored them in the block are called **miners**.

Someone can tempered data in one block. It can change the chain, and it would be immediately apparent it had tampered. If hackers wanted to alter the data in a blockchain system, they need to improve each and every block in the chain over all the string's in distributed versions. Blockchains, Bitcoin and Ethereum are constantly and continually growing as blocks added in the chain, which significantly adds to the protection of the ledger[11].

Blockchain, immutability responsible for its cryptographic hashing, is an essential part of a blockchain. It is liable for providing immutability – the most important features of blockchain technology.

Hashing is a process to produce a fixed-sized output (i.e., SHA1 can produce size

160 bit of hash). No matter whatever size of the input string. The input of the hash function is 1 or 100 characters; the output will always be the same length. The properties of the cryptographic hash function are:

- **Deterministic:**
  A specific input gives to cryptographic hash function much time, but it always produces the same output.

- **Irreversible:**
  It is not easy to reverse the process means we can determine the input from the output.

- **Collision resistance:**
  the two different inputs can not produce the same output.

In a cryptographic hash function, we change the bit of the data; it can change the whole output of this function. For example, the output of the these two string "hi" and "hello" is 0x730e209bd7a8a32b1cb9d9a09aa2325d2430587ddbc0c38bad911525 and 0x 619cba8e8e05826e9b8c519c0a5c68f4fb653e8a3d8aa04bb2c8cd4c respectively. Bitcoin uses SHA-256, and Ethereum uses keccak256 as a cryptographic hash function.

The cryptographic hash function is responsible for immutability in blockchain technology, each block of data contain the hash of their respective previous block of data.

Cryptographic hashing is responsible for immutability in blockchain technology. Each new block of data includes all the data in the previous block as the hash function. i.e., if we added the 100th block in the blockchain. It contains data of the 99th block in the 100th block as hash output. However, block 99's data has a hash of the block 98 and goes on up to the genesis block shown in the fig.

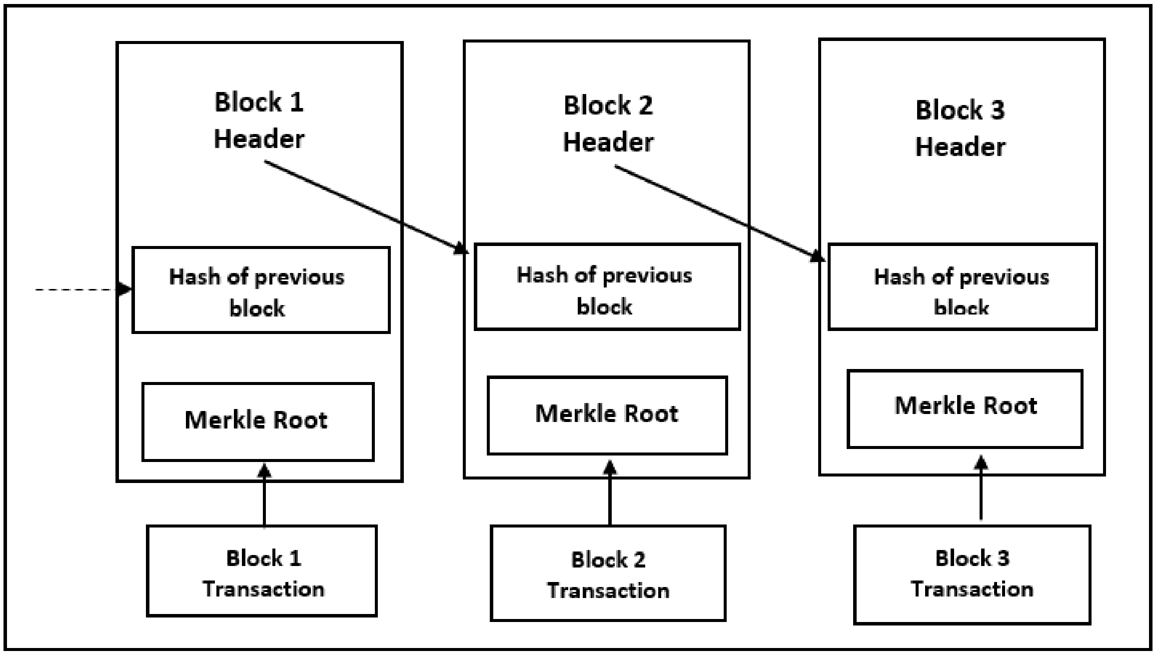

Figure 2.2: Simplified Blocks in the blockchain[15]

These figures show the simplified version of a blockchain that gives us some crucial information. These arrows indicate that every block's transactions are recorded in a Merkle root, and the root of the node is the node of the Merkle tree.

Merkle tree:
Merkle tree is used for the cryptographic hash function to store the hash instead of original data in each node. Every leaf node stores the hash value of their original data, and every parent node hash is a combination of their original data and their child hashes.

The Merkle tree's root node is known as a Merkle root; it represents a hash as they combined all chid hash of their left subtree as well as a right subtree. The below figure shows the Merkel tree:

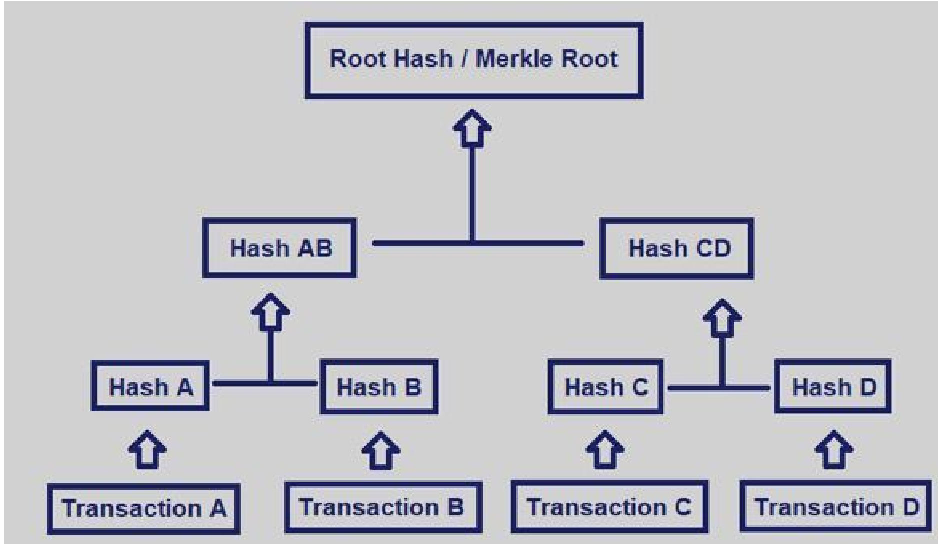

Figure 2.3: Merkle Tree[4]

The leaf nodes consist of the hash of the transaction A, B, C, and D. Then, the hash A and B are combined and produced a new hash as AB also hash C and D produce the hash CD. After that the hash AB and CD are combined and produced the new hash which stored in the Merkle root.

Merkle root is important for blockchain because its cryptographic properties apply to tell that the transaction in a block is altered from the original transaction[4]. If the single transaction changes in the block, then the Merkle root was completely different from the original Merkle root, so we can notice the changes. For example, the transaction in the blockchain using a Merkle tree structure shown in the figure.

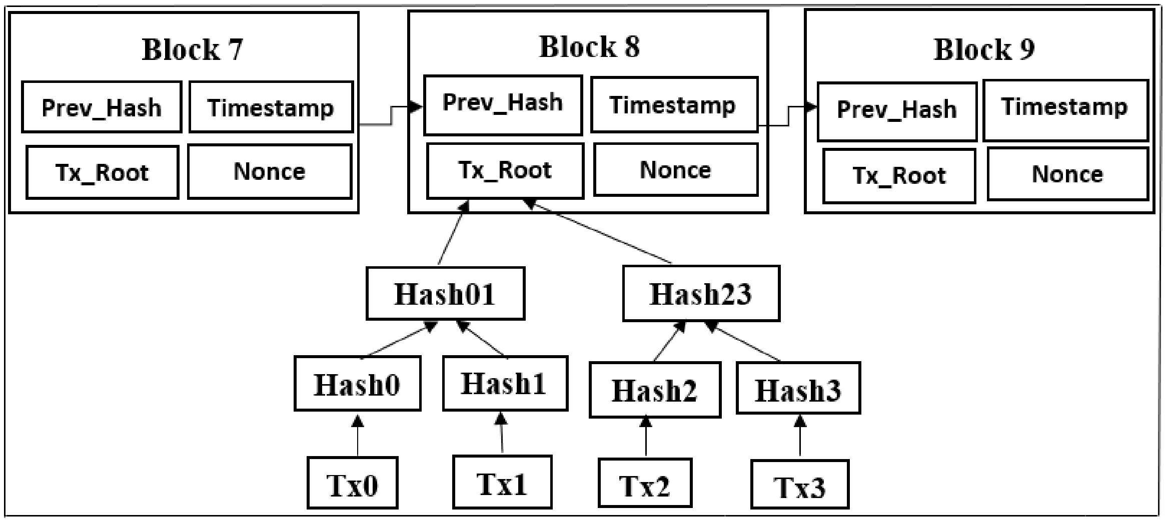

Figure 2.4: The blockchain transaction use the Merkle tree[4].

Merkle tree has the property that the user can verify their transaction in a block without downloading the whole blockchain network. The SPV (Simplified Payment Verification) process traverses the subtree in the Merkle tree and checks that certain transaction hash is in the tree. The blockchain has this level of efficiency because of the Merkel tree.

There are two types of the blockchain network:

- **Public Blockchain:**
  A Public Blockchain is also known as a permissionless blockchain. Anyone can join the blockchain network, signifying that they can read, write, or engage with a public blockchain. Public blockchains are decentralized[7], no one has the authority of the system, and they are defended in that the data can't be modified once validated on the blockchain.

- **Private blockchain:**
  The private or permissioned blockchain or Enterprise Blockchains allows companies to create and centrally administer their transactional networks, inter- or intra-company can use with partners. Enterprises need to ensure security, privacy, compliance, performance, and many of the properties that a private blockchain can provide[7].

The blockchain enables to make the database completely decentralized. This way of data distribution increases its quality, safety, and resistance to restraint. For a long period of time, blockchain was a domain of cryptocurrencies like a Bitcoin, Ethereum, Litecoin, and other cryptocurrencies. Because of these features, the technology is to provide the best protection for financial and identity data. So that many companies turn their business to the blockchain network because it has some properties which help to reduce the cost, easy to trace the transaction and database is decentralized.

Reducing costs: The blockchain is impacting operational costs. By eliminating the party between the server and the data, the access is more convenient and quicker. Blockchain technology improves the speed and effectiveness of processing the data. The process is automated, making it less time-consuming. Also, it decreases the risk of human fails, which may alter the traditional method. That means less financial loss.

Easier tracing: This feature may be especially crucial for the company if it distributes the products based on a complex supply chain. Tracking all the transaction[7] will be more straightforward for the company if they use the blockchain technology. If all the data is put on the distributed ledger, it's easy to examine all the transactions' history. It may be advantageous in terms of increasing security measures.

2.2 Security Attacks in Blockchain Technology

Blockchain, immutability, distributed identity, built trust, distributed consensus, and continuous valid claims may seem like the ultimate security reliable technology. But some security attacks are rising, which are very complex and can cause huge irremediable damages.
The blockchain security attacks can categorized in four categories[7]:

- Peer to peer network-based attacks
- Ledger and Consensus based attacks
- Smart contact based attacks
- Wallets based attacks

2.2.1 Peer to Peer Network-based Attacks

Eclipse attack:

A node will depend on the $'x'$ number of nodes selected using a Peer selection approach to have its way of the distributed ledger. But if an attacker can control to make the node choose all the $'x'$ number of nodes from his malicious nodes alone, he can exceed the original ledger's view and present his own formed ledger to the node.

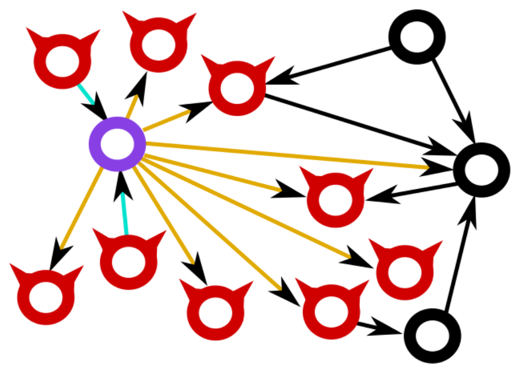

Figure 2.5: Eclipse attacks on peer to peer network[8]

2.2.2 Ledger and Consensus-based Attacks

- **51% attack**
  This attack happens when a miner or a group of miners hold 51% or more of the blockchain network's mining power. Though it is difficult for large systems, the probability of a 51% attack probability is high in small networks. Once a group has control of large part over transactions on a blockchain network, it can stop a specific transaction or even rearrange older transactions.

- **Mining malware:**
  Malware uses to compute the power of native victims' computers to mine cryptocurrencies for hackers. China published that over a million computers were affected by this malware and improved attackers mine more moreover the 26 million tokens of different cryptocurrencies.

- **Timejack attack:**
  Nodes in individual blockchain networks like Bitcoin depend on internal timing derived from the median time informed by its peer nodes. For example, if you are dependent on your friends to know the time. The attacker is contrived to put a group of malicious people in your friends' list, and

then be able to change your time. The first step of this attack is an Eclipse attack on the target node. Once this attack is happening on a target node, then the target node can not accept blocks from the original network as the timestamp of the blocks will not be in order with its timestamp. That provides a chance for the attacker to double-spending or do transactions with the targeted node, and the original blockchain network can not obey these transactions.

2.2.3 Smart Contract-based Attacks:

Smart contracts are fully automated contracts that perform transactions in an agreed-upon process between participants, with the input from the users and without interference from any middlemen. Therefore once started, a smart contract cannot prevent. The transaction is completed and recorded into blockchain. It is immutable. That gives a guarantee to peoples; they can proceed based on their performance. But imagine what happens when the Smart Contract has bugs: Millions of dollars are in stake, and no one can alter it. The DAO attack is smart contract based attack.

2.2.4 Wallets-based Attacks:

Parity Multisig Wallet Attack:
This case of vulnerability is where the attacker hacked the parity client wallet and held up 500,000 Ether. Wallet contracts are produced the additional logic for precise automated payments on user wallets, to overcome transaction fees. The parity Multisig wallet functionality used a centralized Library contract. But they ignored some crucial functions open, resulting in a vulnerability, which the attacker employed. The attacker builds his account as owner in the library contract, so for all wallets developed after a particular date, the attacker now a mutual owner. Then he triggered a kill function, which halted the currencies in the wallet. He barred $ 155 million as of that day, forever in cryptographically unobtainable wallets.

CHAPTER 3
Literature Survey

In this chapter, we will briefly explain how to modify current blockchain technology to obtain a redactable blockchain. We start with a short description of blockchain abstraction due to Garay, Kiayias and Leonardos[12] in section 2.1. In section 2.2. we recall the concept of the chameleon hash function.

3.1 Redactable Blockchain

Blockchain carries two essential aspects of the idea of decentralization and complex smart contracts. Once content or block publish in a chain, then that cannot modify or erase the data in blockchain technology[17]. Immutability is significant to achieve data integrity in a trustless distributed network. 100% immutability cannot theoretically promise in a blockchain environment, proof-of-work and proof-of-stack (strong economic practices) gives practical assurance that only one shared ledger is maintained. Validators are fixed in a private blockchain platform (ripper), easy to achieve immutability[22].

The primary purpose of the design a blockchain for carrying out that financial transaction they did not need for reversing a transaction. Blockchain has immutable properties. They cannot allow modifying the transaction, but in the financial sector require to modified or erase the data, coding, and transaction mistake. This senior unalterable record is not suitable for some applications.

Content can modify or remove with full transparency and accountability to make blockchain redactable. Reduction means compress any number of blocks into a smaller number of blocks, rewrite one or more blocks. A reduction made by the authorized entity under specific constraints.

Redactability in blockchain can be achieved by using the chameleon hash function, it's also called trapdoor hash function. Which can recreate matching algorithms through the use of trapdoor key. After a change has been made to a block, the original blockchain remains fully intact and there is no need to create a hard fork and rebuild subsequent blocks.

The redactable blockchain invention provides the means to build a virtual padlock on the link connecting two blocks (see Figure 3.1). Redacting the blockchain is simple: the chameleon hash key is used to unlock the link between the block that must be changed and its successor, using this key it possible to substitute the block with a new one without breaking the hash chain.

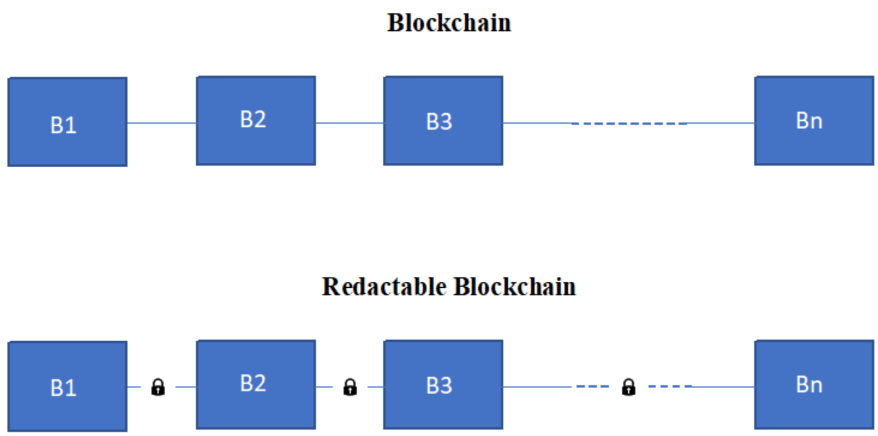

Figure 3.1: Redactable Blockchain[3]

The invention is designed to preserve the virtues of immutability as well. To positively identify blocks that have been changed, it is possible to architect blockchain so that any redaction leaves an inevitable "scar" that cannot be removed, even by trusted parties. These are accomplished by including both a redactable chameleon hash to connect the blocks alongside a standard hash. So while the redactable blockchain capability will not force a node to purge data from their archives, users will now have the technical ability to comply with privacy laws.

A successful prototype of the invention was created by modifying bitcoin's core technology, which is the most widely used blockchain technology. The design modifications are possible in a range of existing blockchain technologies, requiring only minimal and inexpensive changes to the current blockchain, block, or transaction structures and how local participant software interprets the information.

3.1.1 Blockchain Basics:

We use the notation of [12] to describe the blockchain. A block is triples of the $B =< s, x, ctr >$, where s $\in \{0, 1\}^*$, ctr $\in$ N.
Block B is valid if;
$validblock_q^D$(B) := (H(ctr,G(s, x))< D) $\wedge$ (ctr< q) = 1.

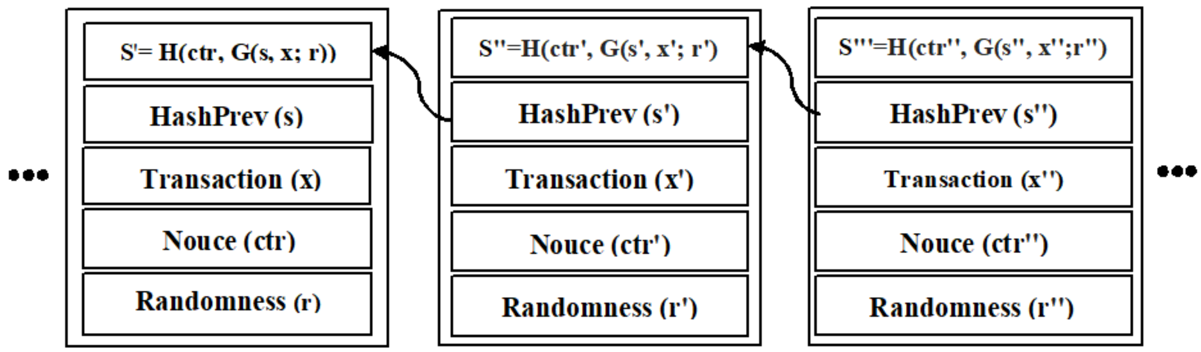

Figure 3.2: The redactable blockchain structure (using a public-coin chameleon hash)[3].

The field s of a block stores the value shown in the top white field of the previous block. We note that the top white field is not stored in the block. The bottom darker field Randomness) is updated when the block is redacted (a collision is computed).

Here, H : $\{0, 1\}^* \rightarrow \{0, 1\}^k$ and G : $\{0, 1\}^* \rightarrow \{0, 1\}^k$ are collision-resistant hash functions, and the parameters D $\in$ N and q $\in$ N are the block's difficulty level and the maximum number of hash queries that a user is allowed to make in any given round of the protocol, respectively.

The blockchain is simply a chain of blocks that call C. the rightmost block is called head of C. The head(C) :=<s, x, ctr> can be extended to a new longer chain (C'):= C|| B' by attaching the valid block $B' = <s', x', ctr'>$. The chain also can be empty C = $\in$. The function len(C) denotes the length of a chain C (i.e., total blocks in chain). For a chain C of length n and any k $\geq$ 0, we denote by $C^{\lceil k}$ the chain resulting from removing the k rightmost blocks of C, and analogously we denote by $^{\lceil k}C$ the chain resulting in removing the k leftmost blocks of C; note that if k $\geq$ n then $C^{\lceil k} = \in$ and $^{\lceil k}C = \in$. If C is a prefix of C' we write $C \prec C'$. We also note that the difficulty level D can be different among blocks in a chain.

This applies to public or permissioned blockchain where only specified actor (like banks, financial operators, individuals, etc.) participate and post the transaction. We focused on PoW for simplicity, but PoW does not make sense in a pure permissioned blockchain where other consensus protocols are used (e.g., PBFT)[11]. A solution that depends on the consensus mechanism is not desirable.

3.1.2 Reduction in Private Blockchain:

The Private Blockchain is also known as a **permissioned blockchain**. Permissioned networks place limitations on who is permitted to participate in the system and the type of transactions. The first idea following our approach is to set the inner hash function (function G), used to chain the different blocks in the blockchain, to be a chameleon hash function. Intuitively, re-writing the content of each block is possible by finding collisions in the hash function (without modifying the outer hash function H).

The chameleon hash function's trapdoor key should be given to the centralized authority to calculate collisions and accordingly redact blocks. We require some modifications to the earlier defined block. A block is now a tuple B : =$\langle$s, x, ctr, (h, ξ)$\rangle$, where the elements s, x and ctr are the same as earlier, and the aspect (h, ξ) is the hash and check string pair for a chameleon hash. The function G is defined to be a secret-coin (standard chameleon hash function) chameleon hash CH=(HGen, Hash, HVer, HCol), and the validation predicate for a block is:
$Validblock_q^D$(B) := H(ctr, h) $<$ D) $\wedge$ (HVer(hk, (s, x), (h, ξ))) $\wedge$ (ctr $<$ q)=1.

We have no need to store hash value because it is determined as the randomness of the chameleon hash function and the deterministic function's input. There is a case where block B : = s, x, r, ctr, where r is the randomness of the chameleon hash function. The validation predicate for block
$Validblock_q^D$(B) : = H(ctr, Hash(hk, (s, x); r)) $<$ D) $\wedge$ (ctr $<$ q)=1.

3.1.3 Reduction in Public Blockchain:

A Public Blockchain is also known as a **permissionless blockchain**. Anyone can join the blockchain network, meaning that they can read, write, or participate with a public blockchain. The users easily access the blockchain on bitcoin and ethereum. The public blockchain contains a significant factor that no one can obtain the information which is available on the blockchain. No one has the authority to change the blockchain protocol and the information available on the blockchain.

The major disadvantage of the public blockchain is that the miner and computer are connected in the network, so that it needs a large number of computational power also all miners in a distributed ledger maintained by a considerable scale. Each node included in the network can go through the proof of work, which can help to include consensus integrity to solve the cryptographic problem, and this is synchronized with other miners. Another drawback is that the public blockchain can not ensure the privacy and security notation. All transactions in The public blockchain are publicly visible for users and other miners; they are inclined to eavesdropping or thwart hacking[2]. In a decentralized network, each miner is their in way that the blockchain can only be accessed by a the private key(trapdoor key). Public blockchains are decentralized, here no one has control over the system, and they are secure in a way that the data can't be changed once validated on the blockchain. The trapdoor key is managed in two ways:

1. The trapdoor key can be divided among all the people (miners) of the network. The disadvantage of this solution is that if the number of people in the system is vast (> 200), it might not be entirely sufficient due to the MPC protocol's performance concerns.

2. The trapdoor key can be shared with a carefully chosen subset of the peoples. For example, in Bitcoin, it is quite identified that the preponderance of the network hashing power is actually managed by a small number of people (the top 7 mining pools control around 70% of the network total hashing power). Although we recognize that the thought of hashing power to a small number of peoples can be toxic to the system, this solution does not replace the existing Bitcoin trust assumption (Bitcoin already considers as trusted majority).

3.2 Chameleon Hash Function

Cryptographic hash function (like SHA1, MD5, . . .) is a one way function mean it is infeasible to invert; this is pre-Image resistance properties if we want to calculate input x from their hash value z, only one-way using brute force search of possible inputs to see if produce a match. Second, pre-image resistance properties it is hard to find that two different inputs have the same hash value. The finding a collision in SHA1 is 263. Collision Resistance is made very hard for an attacker to find the same hash value for two different inputs; if the cryptographic hash function is collision-resistant, then it also second pre-image resistant.[1]

The collision-resistant hash function maps an arbitrary message to a hash value, and if we use SHA1, then the hash length is 160bits, which is enough to construct the hash element of the chameleon hash function. This fact is used to implement the chameleon hash function and also used in implementation of their application. The chameleon hash function directly applies to any arbitrary message, but it can give faster and more efficient hash value as a result if it is applied to any regular cryptographic hash function then applying the chameleon hash function. The forceful contraction of a chameleon hash function is based on the assumption of the standard cryptographic hash,the chameleon hash function can efficiently contract factorization[14] and discrete logarithm[5].

In the implementation of chameleon hash, we use the trapdoor commitments introduced by Brassard, Chaum, and Crepeau[9] in a context of zero-knowledge proof[14]. The non-interactive commitment and chameleon commitment scheme causes a chameleon hash function.

A chameleon hash function is non-standard type of collision resistant hash function,which is associated with a pair of hash key (public key) and trapdoor key (private key), and it is a collision-resistant hash function which uses notation of chameleon commitment[14].

Trapdoor key knowledge uses to generate hash collision efficiently without knowledge of trapdoor key hard to find a collision[6].

- Those know the public key can generate the associated hash function.
- Those who do not know the trapdoor key then it is infeasible to find two inputs that mapped the same output.
- However, the owner of the trapdoor information can easily find a collision for each given input.

The Chameleon hash function is associated with user who has public key(h_{hk}) and private key(trapdoor key)(tk). The pair of keys is generated by user using key generation algorithm.

The public key h_{hk} denes a chameleon hash function, denoted $\mathrm{CH}_R(. ; .)$, which can be computed eciently given the value of $\mathrm{h}_h k$. On input a message m and a random string r, this function generates a hash value CH_R(m; r)[6] which satisfies the following properties:

Collision resistance:
There is no efficient PPT (Probabilistic polynomial time) algorithm can find the pairs of (m_1, r_1) and (m_2, r_2) where $m_1 \neq \mathrm{m}_2$ such that $CH_R(m_1, r_1) = \mathrm{CH}_R(\mathrm{m}_2, \mathrm{r}_2)$, except negligible properties.

Trapdoor collision:
There is efficient algorithm that on input the trapdoor key, any pair $\mathrm{m}_1\mathrm{r}_1$, and any additional message m_2, find a value r_2 such that $\mathrm{CH}_R(\mathrm{m}_1, \mathrm{r}_1) = \mathrm{CH}_R(\mathrm{m}_2, \mathrm{r}_2)$.

Uniformity:
All messages m cause the same probability distribution on CH_R(m; r) where r uniformly chosen at random. This condition can be relaxed to require that the above distributions are not necessarily identical for all messages but computationally indistinguishable.

The chameleon hash function is classified in non-standard property of collision-resistant for signer(sender) and collision tractable for receiver[1].

The chameleon hash function is collision resistance hash function unless trapdoor is not known. The knowledge of trapdoor allows finding arbitrary collision in the domain of the function. Secret-coin hashing is a standard concept of the chameleon hash function[10][3].

A chameleon hash function is defined by a set of efficient algorithms; CH(HGen, Hash, Hver, Hcol) specified as Follow:

Key Generation algorithm(HGen): It takes input as security parameter k in unary form, and outputs a pair of a public key and trapdoor key (h_{hk}, tk). It is a probabilistic algorithm.

Key Generation : $1^k \rightarrow$ (h_{hk}, tk).

Hash: It takes input as a public key, message m, and implicit uniformly distributed r, and output is a pair of the hash value and check string ξ.
Hash : (h_{hk}, m, r) $\rightarrow$ (h, ξ).

Hash Verification(Hver): It takes input as public key h_{hk}, message m, and pair of the hash value and check string, and output is bit value 0 or 1 if the function returns 1, then the generated pair of the hash value and check string for particular message m is valid.
Hash Verification: (h_{hk}, m, (h,ξ)) $\rightarrow$ {0, 1}

Collision Finding algorithm(Hcol): It takes input as trapdoor key, valid tuple(h,m, ξ), and a new message m$'$ and return the new string ξ'. So that
Hver(h_{hk}, m , (h, ξ)) = Hver(h_{hk}, m$'$, (h,ξ')) = 1

The secret coin hashing algorithm is randomized on the input message m. It can produce the hash h and check string ξ, and help in verify the hash key to recomputing the hash. The hash algorithm for a random coin is secret. A specific case where the check string ξ contains the random coin r, which was used to generate h, so when m and r are fixed, the hash computation is completely deterministic. It is called **the public coin chameleon hash algorithm**. Here the verification algorithm merely is the hashing algorithm so that we can drop the verification algorithm in the case of the public coin from the chameleon hash.

The public coin chameleon hashing contain four tuples same as secret coin hashing CH = (HGen, Hash, HVer, HCol) but there are some differences in Hash generation algorithm and verification algorithm:

- **The hash generation algorithm** takes input as a public key(hash key) h_{hk} and message m and the return output as a pair (h, r), where r $\in$ R_{hash}, an implicit random coin which was used to produce the hash value.

- **The hash verification algorithm(Hver)** takes the input as a message m, public key h_{hk}, and the pair (h, r), and they return the output as one if it is valid. It was possible only when, Hash(h_{hk}, m; r) = h.

If we implement digital signature using the chameleon hash function, this schema is known as **chameleon signature**, which follows below written properties:

- **Unforgeability:**

There are No any third party can provide an (R, S)-proper triple SIG(m) = (m, r, sig) not earlier generated by the signer S. The receiver R can provide an (R, S)-proper triple (m, r, sig), particularly for those values sig earlier provided by S.

- **Non-transferability:**
  Except for the signer, no one can able to prove to the other party that the signer provided a given triple SIG(m) = (m, r, sig) for any such triple. That should be true for the recipient and for any third party (say a judge)(that participated in a rejection or confirmation protocol with the signer).

- **Denial:**
  In case of conflict, if the signer S is displayed with a triple SIG(m) = (m, r, sig) not provided by him, then S can prove an (honest) judge J to reject SIG(m).

- **Non-repudiation:**
  If there has a case of conflict, then the signer S is displayed with a triple SIG(m) = (m, r, sig). Which is created by him, then S cannot convince an (honest) judge J to reject SIG(m).

- **Exposure free:**
  A chameleon signature scheme is vulnerability free if the signer can refuse a false signature (triple (m, r, sig) not written by him) without revealing any other message actually signed by him.

The chameleon hash function establishes the security and considers the vulnerabilities against attacks[18]. If the attacker breaks the function, then the secret key would recover through the third party, which allows the attacker to get information about the trapdoor, which is associated with an identity string. If the attack is less powerful, then it extracts some identity string and trapdoor information; selected by the attacker. However, these types of attacks are dangerous.

- **Random message attack:**
  The attacker can reach an oracle and signs the arbitrary message; chosen by the oracle.

- **Generic chosen message attack:**
  The attacker is provided signatures are provieded to the attacker for a list of messages of his choice. This list of the message can not produced before any signature on it, which is independent of the verification key.

- **Adaptive chosen message attack:**
  The attacker has the authority to sign a queried message m in oracle. Each query m depends on the trapdoor key tk which was used to produce the signature for the previous message.

- **Q-adaptive chosen message attack:**
  The attacker can query to the oracle at most Q time for a specific message.

CHAPTER 4

Proposed Work

In this Chapter, we will be focusing on characterization of chameleon hash function based on their implementation using standard hash function that we worked upon. We have implemented the approaches discussed here, and compared their results with the results obtained for our proposed approach.

Chameleon Hash Function

Chameleon hash function is characterized based on non-standard properties in Simple Factorization, Discrete Logarithm and Advanced Factorization.

4.1 Trapdoor Function Based on Simple Factorization (Claw-free Permutation)

Claw-free permutation: A pair of permutations ($f_0(x)$, $f_1(x)$) over a common domain is called **"claw-free"** if computationally infeasible to find values x and y from the domain such that $f_0(x)$ = $f_1(x)$. Using these two functions, we can compute claw by choosing x randomly.
For example take RSA permutation pair here,
$f_0(x)$ = x^3 (mod n) and $f_1(x)$ = x^5 (mod n)
Therefore, gcd $\phi(n)$ = 15
$Z \equiv f_0(x) \equiv f_1(y) \equiv x^{15}$(mod n).

Using factoring, it is difficult to generate pair of claw.
The set of composite numbers which are the product of two k bit primes which are both congruent to 3(modulo 4) but not congruent to each other modulo 8.

$H_k = \{n = pq \mid |p| = |q| = k, p \equiv 3(\text{mod } 8) \text{ and } q \equiv 7(\text{mod } 8)\ \}$

Define claw-free permutation functions using factorization algorithm
$f_{\{0,n\}}(x)$ = {x^2(mod n) if x^2(mod n) < n/2, $-x^2$(mod n) if x^2(mod n) > n/2 }
$f_{\{1,n\}}(x)$ = {$4x^2$(mod n) if $4x^2$(mod n) < n/2, $-4x^2$(mod n) if $4x^2$(mod n) > n/2 }
In factorization, generating inverse properties are stronger than generating collision. Hence, factorization can convert any signature scheme to secure signature scheme, which is secure against random message attack.

Key Generation algorithm:
Randomly Choose two primes p, q $\in (0,1)^{k/2}$ where k is security parameter, such that p = 5(mod 8) and q = 1(mod 8) and n = pq, where n is hash key hk and (p, q) is trapdoor key tk. Both of these keys are implemented in `CS_GenChamKeys_SF()` function.

Hash Generation algorithm:
$f_0(x) = x^2$(mod n), $f_1(x) = 4x^2$(mod n) and the domain of this function is D = { x $\in Z_n^*$ | (x/p) = (x/q) = 1 }. If message M = m[1]m[2]. . . ..m[k] and randomly chosen r $\in Z_n^*$ then hash h_{hk}(m;r) = $4^m r^{2|k|}$ (mod n). `CS_ChamHash_SF()` function returns h_{hk}(m; r) as a hash value.

Trapdoor Collision:
The value of m_1, r_1, message $m_2 \in M$ and $h_{hk}(m_1, r_1)$ are given. Using these values, we can get r_2 with help of given function:
$r_2 = f^{-1}_{m2[1]}(f^{-1}_{m2[2]}(\ldots\ldots(f^{-1}_{m2[|m2|]}(h_{hk}(m_1; r_1))$
This results in $h_{hk}(m_1, r_1) = h_{hk}(m_2, r_2)$. Here, function (f_0^{-1}, f_1^{-1}) is computed in polynomial time, therefore the value of r_2 is also computed in polynomial time by `CS_ClsnFind_SF()` function.

Chameleon hash function based on the simple factorization is satisfies below property:

- **Efficiency:**
  The hash key hk = n and a pair (m; r) $\in$ M X QR_n, the function h(m; r) = $4^m r^{2|k|}$ (mod n) can be computed in polynomial time.

- **Collision resistance:**
  Assume to the contradictory, that there is exist a probabilistic polynomial time algorithm that gives a hash key n and the outputs pairs (m_1; r_1), (m_2; r_2) $\in$ M X QR_n such that $m_1 != m_2$ and $h_{hk}(m_1; r_1) = h_{hk}(m_2; r_2)$, with a probability that is not negligible. Let take the smallest index of as a bit where m_1

and m_2 vary (where $m_1[j] = m_2[j]$ and $m_1[i] \; ! = m_2[i]$ for every $j < i$). Such a bit endures due to the suffix-free property of M. As we arrogate that the hash function results on (m_1; r_1) and (m_2; r_2) are the same and the $m_1[j] = m_2[j]$ for each $j < i$, and (f_0; f_1) are permutations, it follows that
$f_{m_{1[i]}} o ... o f_{m_{1[|m1|]}}(r_1) = f_{m_{2[i]}} o ... o f_{m_{2[|m2|]}}(r_2)$
Thus, we found a pair of values r'_1 and r'_2 for which $f_{m_{1[i]}}(r'_1) = f_{m_{2[i]}}(r'_2)$. As proven in the existence of such claws for $(f_0; f_1)$ contradicts the Simple factorization.

4.2 Trapdoor Function Based on Discrete Logarithm

Key Generation algorithm:
Randomly choose large primes p and q, where p = kq+1 and $g \in Z_n^*$ of order q. Choose random variable $x \in_R Z_q^*$ and compute hash key $y = g^x (\text{mod } p)$ where (p, q, g) is implicit part of hash key and x is trapdoor key. Both of these keys are implemented in `CS_GenChamKeys_DL()` fuction.

Hash Generation algorithm:
`CS_ChamHash_DL()` function returns hash value
$h_{hk}(m; r) = g^m y^r (\text{mod } p)$, where m is the given message and r is the random variable $r \in Z_q^*$.

Trapdoor Collision:
Given pair $(m_1, r_1) \in Z_q \times Z_q$ and message $m_2 \in Z_q$, we get $r_2 \in {}_q$ using following equation,
$r_2 = x^{-1}(m_1 \breve{\ } m_2) + r_1 (\text{mod } q)$.
Here, r_1 is uniformly distributed variable. So, r_2 is also uniformly distributed variable in Z_q. `CS_ClsnFind_DL()` function returns the value of r_2.

Chameleon hash function based on the discrete algorithm is satisfies below property:

- **Efficiency:**
  The hash key hk = (p, g, y) and a pair $(m, r) \in Z_q \; X \; Z_q$, the function $h_{hk}(m, r) = g^m y^r$ (mod p) are computable in polynomial time.

- **Collision resistance:**
  Assume to the contrast that there is a probabilistic polynomial-time algorithm that given a hash key hk = (p, g, y) and the output pairs (m_1, r_1), (m_2, r_2) $\in Z_q$ X Z_q so that $m_1 != m_2$ and h_{hk}(m_1, r_1) = h_{hk}(m_2, r_2), with a hazard which does not mean negligible. The y is a discrete log with respect to the base g could be determined in polynomial time from the output, as results. Let express the discrete log of y. Then
  $m_1 + \alpha$ r_1 = $m_2 + \alpha$ r_2 (mod q)
  The fact that $m_1 != m_2$ (mod q) implies that $r_1 != r_2$ (mod q), and thus r_1 - r_2 is invertible modulo the prime q. Hence, α can be determined in polynomial time as follows.
  α = $(r_2 - r_1)^{-1}(m_1 - m_2)$ (mod q).

4.3 Trapdoor Function Based on Advanced Factorization

Key Generation algorithms:
Choose two prime numbers p, q $\in$ $(0,1)^{k/2}$ and compute n = pq. Choose random variable g $\in Z_n^*$ of order $\lambda(n)$, where ($\lambda(n)$ = lcm(p-1, q-1) = 2p′q′) and p = (p-1)/2, q = (q-1)/2. Here, (n, g) is pubic key and (p, q) is private trapdoor key. Both of these keys are generated in `CS_GenChamKeys_AF()`.

Every r is uniformly distributed in Zλ(n); therefore, a polynomial number of signatures unveil a logarithmic number of the standard meaningful bits in the secret λ(n). But, this is not vulnerable until the known n, and the secret ϕ(n) = 2λ(n) has the identical bits in their top halves.

Hash Generation algorithms:
Hash h_{hk}(m; r) = g^{mor}(mod n). where **mor** is denoted as concatenation of m and r. Where m is the message and r is the random variable r $\in Z_n^*$. `CS_ChamHash_AF()` function returns h_{hk}(m; r) as a hash value.

Trapdoor Collision:
Hash key hk = (n, g) and pair (m_1, r_1) $\in Z_q \times Z_{\lambda(n)}$ and additional message $m_2 \in Z_q$. We can get r_2 using following equation,
$r_2 = 2^k(m_1 - m_2) + r_1(\text{mod } \lambda(n))$ $r_2 \in Z_\lambda(n)$.

r_2 is generated using `CS_ClsnFind_AF()` function.

The complexity of collision finding in Advanced factorization is equivalent to one-tenth of a regular modular multiplication for 1024 bit keys and 160 bit (hashed) messages, and it requires only two additions or subtractions and one conversion of an 1184 bit number modulo a 1024 bit number.

Chameleon hash function based on Advanced factorization satisfies below property:

- **Efficiency**
  The hash key hk = (n, g) and pair (m, r) $\in Z_n$ X $Z_{\lambda(n)}$, the function h_{hk}(m,r) = g^{mor} (mod n) is computable in polynomial time.

- **Collision resistance:**
  The PPT algorithm on the input hk = (n, g) and the output pairs (m_1, r_1) = (m_2, r_2) $\in Z_n$ X $Z_{\lambda(n)}$ so that $g^{m_1 \circ r_1}$ = $g^{m_2 \circ r_2}$ (mod n) with probability which is not negligible.
  X = (m_1 o r_1) ($m_2 \circ r_2$) (this equality is over Z). x $!= 0$ and m_1 $!= m_2$. The fact that g^x = 1 (mod n) implies that λ(n) divides x.Thus, $\phi(n)$ divides 2x(ϕn) = (p1)(q1) = 4p′q′ = 2λ(n)). Therefor, there exists a probabilistic polynomial time algorithm, that on input (n, g) and the outputs is a multiple of ϕ(n). It is known that from any multiple of ϕ(n) the factorization of n can be effectively calculated. So we can found that probabilistic polynomial time algorithm that help to solves the Factoring Problem with a probability which is not negligible.

Implementation of chameleon hash functions in SF, DL, AF

Experimental Setup:
GNU/Linux (i486) platform is used to carried out the implementation, with GCC-3.3.5, OpenSSL 0.9.7e[19] library for cryptographic primitives (without any external cryptographic acceleration) and numerical analysis. We haven't used any code optimisation of GCC while building our executable to get any kind of fair computational estimation.

Measure Execution Time

Different ways are possible to examine the process execution time. We applied start_counter() and get_counter() method for tracing the CPU cycles consumed throughout the program execution. The AMD 750MHz machine is used for executing the experiments. It complies with IA32 architecture (which gives cycle counter; a 64-bit, unsigned number). The IA32 counter resides with the read time stamp counter instruction. This instruction does not take arguments. The register %edx is set to the 32 bits high-order of the counter and the register %eax set to the 32 bits low-order. Based on their methodology, a combined pair of functions with the code allows us to calculate the total number of cycles that pass between two-time points:

void start_counter(); /* The counter starts counting cycles */
double get_counter(); /* It returns the total number of the cycle up to the last call of start_counter */

This approach's accuracy verifies using the counter before and after sleep (sleep time) of the function call (i.e., sleep time equals one). We took 756,154,624.0 as return value (756.2 MHz). We ran each function of our interest for 1001 times and repudiate the first values performance time in preference of the cache warming process. Moreover, the results are concluded in run-level 1; to minimize intervention from different processes. To frame all the results into graphs on a common scale.

We implemented the Chameleon scheme with three different methods, namely: Simple Factorization (SF), Discrete Logarithm (DL), and Advanced Factorization (AF). The implementation of these schemes results is described into two phases: Hash Generation (HG) and Finding Collision (FC). These designs create a hash of length 160-bits. The results are decreased in the graphs shown in Figure and Table shown below. The results implemented in the following chameleon hash implementations and Table are the averages taken over 1000 runs each.

Algorithm 1: Implementation of chameleon hash functions in SF, DL, AF

```
for i = 0 to chainlength do
    Start_counter( ) function starts counting hash value generation time.
    Generate hash key and trapdoor key using key generation algorithm.
    Generate chameleon hash value for message m1 h_hk(m1, r1) using hash
      generation algorithm.
    Start_counter function's return value get stored in array variable and
      again it start counting time for collision time.
    Get value of h_hk(m_1, r_1), m_1 and m_2 with help of trapdoor collision
      function.
    Hash generation algorithm helps in getting value of h_hk(m_1, r_1).
    if h_hk(m_1, r_1) = h_hk(m_2, r_2) then
        Start_counter() function returns collision time, which is used to
          generate collision and that value gets stored in array variable.
```

Result: This algorithm returns hash generation time and collision finding time.

Those equations are used to find collisions in the first and second trapdoor hash function. It looks like, but they implemented based on the different security assumptions (discrete log. and factoring). This difference makes it possible to replace the multiplication operation α^{-1}(m_1 - m_2) by the single left shift operation 2^k(m_1 - m_2), which frees around half the total time. And also when the size of the modulus is 1024 bits and the size of the (hashed) (m_1 - m_2) is 160 bits, the reduction of the 1184 bit result modulo a 1024 bit modulus is about six times faster than a standard decrease of a 2048 bit product modulo a 1024 bit modulus. Consequently, we estimate that the collision finding procedure's software implementations will be about ten times faster than performing a single modular multiplication of two 1024 bit numbers.

4.3.1 Results

Below graphs shows resulting data for given chain length value.

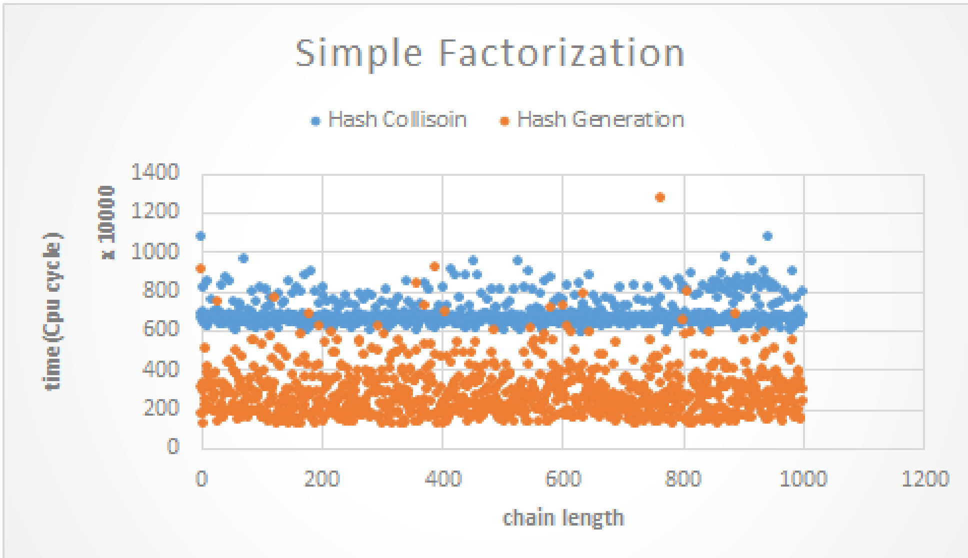

Figure 4.1: Simple Factorization

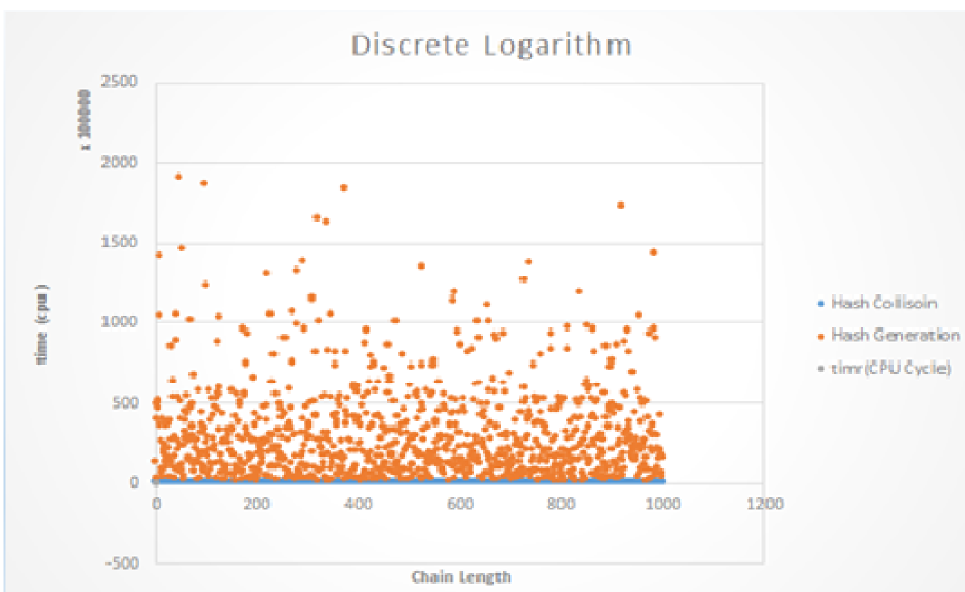

Figure 4.2: Descrete Logarithm

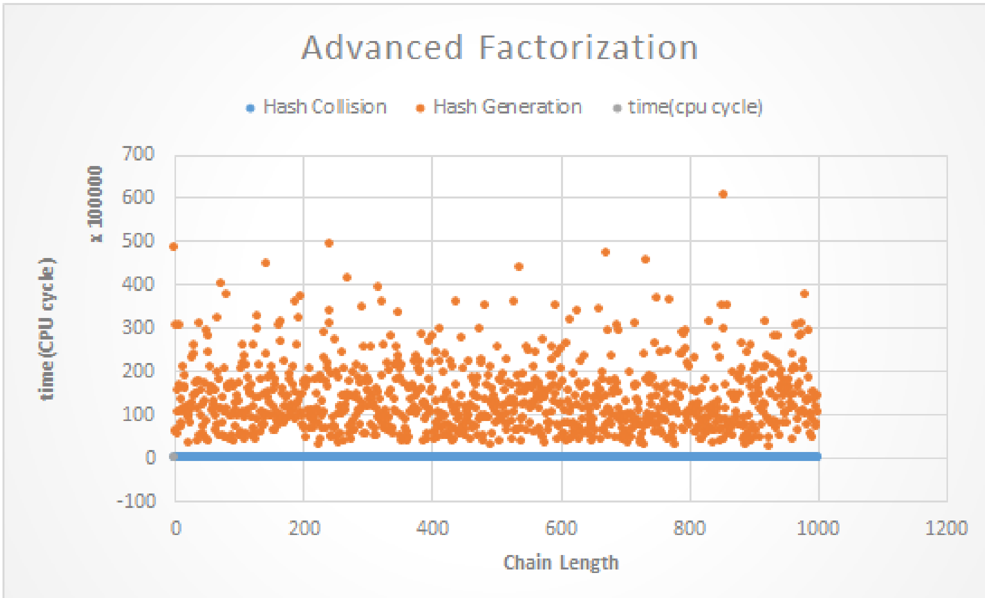

Figure 4.3: Advanced Factorization

Chameleon Hash schemes	**times(ms)**	
	Hash Generation	**Collision Finding**
Simple Factorization	28.90	7.1
Discrete logarithm	334.2	11.46
Advanced Factorization	132.19	69.81

Table 4.1: A-Index for various Indian Languages

Chapter 5
Conclusions

In this thesis, we studied blockchain technology, and their properties also convert into the redactable blockchain proposed by Giuseppe Ateniese, Bernardo Magri, Daniele Venturi, and Ewerton Andrade[8]. It can achieve using the chameleon hash proposed by H. Krawczyk and T. Rabin[1] instead of a cryptographic hash. We have worked on the characterizations of the chameleon hash function. Here we have presented a different implementation of chameleon hash functions; these three variants of chameleon hash function take different time for hash generation and collision finding. The time complexity of the scheme is a probabilistic polynomial-time(PPT). Here, we try to implement the optimal solution of the chameleon hash function. However, it is not as much optimal solution. Its more optimal solution might be possible if we try to reduce or the multiplication operation with the left shift operation to generate the hash value in chameleon hash function schemes.

References

[1] I. B. amgård. Collision free hash functions and public key signature schemes. In D. Chaum and W. L. Price, editors, *Advances in Cryptology - EUROCRYPT' 87*, pages 203–216, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[2] M. Andrychowicz, S. Dziembowski, D. Malinowski, and Ł. Mazurek. On the malleability of bitcoin transactions. In M. Brenner, N. Christin, B. Johnson, and K. Rohloff, editors, *Financial Cryptography and Data Security*, pages 1–18, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[3] G. Ateniese, B. Magri, D. Venturi, and E. Andrade. Redactable blockchain – or – rewriting history in bitcoin and friends. In *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, pages 111–126, 2017.

[4] G. Becker. Merkle signature schemes, merkle trees and their cryptanalysis. In *Chair for Embedded Security*, Ruhr-Universitat Bochum, 2008.

[5] J. F. Boyar and S. A. Kurtz. A discrete logarithm implementation of perfect zero-knowledge blobs. In *Journal of Cryptology 1990 International Association for Cryptologic Research*, volume 2, pages 63–76, University of Chicago, 1100 East 58th Street, Chicago, IL 60637, U.S.A. Mark W. Krentel Rice University, Houston, Texas, U.S.A., 1990.

[6] J. Camenisch, D. Derler, S. Krenn, K. Pöhls, Henrich C.and Samelin, and D. Slamanig. Chameleon-hashes with ephemeral trapdoors. In S. Fehr, editor, *Public-Key Cryptography – PKC 2017*, pages 152–182, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[7] X. B. L. M. e. a. Chen, G. Exploring blockchain technology and its potential applications for education. In *Smart Learning. Environment.*, volume 5, 2018. `https://doi.org/10.1186/s40561-017-0050-x`.

[8] A. Z. S. G. Ethan Heilman, Alison Kendler. Eclipse attacks on bitcoin's peer-to-peer network. Boston University, Hebrew UniversityMSR Israel, 2015. `http://cs-people.bu.edu/heilman/eclipse/`.

[9] D. C. G. Brassard and C. Crepeau. Minimum disclosure proofs of knowledge,. In *Journal of Computer and System Sciences*, volume 37, pages 156–189, October 1988.

[10] t. y. n. . `https://doi.org/10.1016/j.ins.2019.09.001`. v. n. p. G. Mojtaba Khalili, Mohammad Dakhilalian, Willy Susilo, booktitle=Information Sciences.

[11] G. M. D. B. W. R. Gall, Joyce P. Applying educational research: A practical guide. 1999.

[12] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[13] R. L. Karim Sultan, Umar Ruhi. Conceptualizing blockchains: Characteristics applications. In *Computers and Society (cs.CY); Cryptography and Security (cs.CR)*, pages 1–9, 2018.

[14] H. Krawczyk and T. Rabin. Chameleon hashing and signatures, 1998.

[15] V. Lemieux. Blockchain for recordkeeping: Help or hype?, 10 2016.

[16] J. Moubarak, E. Filiol, and M. Chamoun. On blockchain security and relevant attacks. In *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pages 1–6, 2018.

[17] S. M. Shafi Goldwasser and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. In *SIAM Journal on Computing, 1988*, volume 17, pages 281–308, 1988.

[18] A. Shamir and Y. Tauman. Improved online/offline signature schemes. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, pages 355–367, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[19] The OpenSSL Project. OpenSSL: The open source toolkit for SSL/TLS. April 2003. `www.openssl.org`.

[20] S. Voshmgir. *Token Economy: How Blockchains and Smart Contracts Revolutionize the Economy*, volume 2. Shermin Voshmgir, 2019.

[21] D. H. Yum, J. W. Seo, and P. J. Lee. Trapdoor sanitizable signatures made easy. In J. Zhou and M. Yung, editors, *Applied Cryptography and Network Security*, pages 53–68, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[22] F. Zemler. Concepts for gdpr-compliant processing of personal data on blockchain: A literature review. 12 2019.