Quantum-safe Bitcoin

How to protect blockchain systems from quantum-computer attacks

Trung Van Trinh

Thesis submitted for the degree of
Master in Network and system administration
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2020

Quantum-safe Bitcoin

How to protect blockchain systems from quantum-computer attacks

Trung Van Trinh

Acknowledgments

I would like to thank my supervisor Audun Jøsang for helping me with building a good theoretical foundation for this thesis, Audun was always available for me when I need guidance and clearly expressed what I needed to learn for doing this project. This thesis would not have been possible without his help. I am grateful to have Audun as my supervisor.

I also want to thank my parents for supporting me all these years and for making delicious food for me to grow up healthy and strong.

Abstract

In this thesis, we introduce the basics of blockchain systems, cryptosystems, and post-quantum cryptography. We discuss how to upgrade and make digital signatures in the cryptocurrency Bitcoin more robust against quantum attacks. In addition, we propose the hash-based post-quantum digital signature Wintersnitz OTS, and SPHINCS+, and describe their application in post-quantum blockchains.

We came up with two proposals to upgrade the cryptocurrency Bitcoin. The first proposal is to add a post-quantum signature to the Bitcoin ECDSA public key. The second proposal is to iterate the Bitcoin ECDSA public key 256 times, where each of these keys can be used to make a Bitcoin address. In addition, we also discuss the potential vulnerabilities of our two proposals.

Contents

I The First Part **1**

1 Introduction **3**
- 1.1 Motivation and Research Questions . . . 3
- 1.2 Research Method . . . 4

2 Blockchain **5**
- 2.1 History of Digital Money . . . 5
- 2.2 Blockchain Technology . . . 6
  - 2.2.1 What is a blockchain? . . . 8
  - 2.2.2 Blockchain Block . . . 8
  - 2.2.3 Merkle Tree . . . 10
  - 2.2.4 Mining . . . 11
  - 2.2.5 Blockchain Consensus . . . 12
  - 2.2.6 Proof-of-Work . . . 12
  - 2.2.7 Proof-of-Stake . . . 13
  - 2.2.8 Smart Contract . . . 14
- 2.3 Permissionless Blockchain . . . 14
  - 2.3.1 Bitcoin . . . 14
- 2.4 Permissioned blockchain . . . 16
  - 2.4.1 The Interplanetary File System (IPFS) . . . 17
  - 2.4.2 Libra Blockchain . . . 18
  - 2.4.3 Hyperledger . . . 18
- 2.5 Fork on Blockchain . . . 19
- 2.6 Attacks on Blockchains . . . 20

II The Second part **23**

3 Cryptosystems **25**
- 3.1 Introducing to Cryptography . . . 25
- 3.2 Security Concepts . . . 25
- 3.3 Symmetric Cryptography . . . 26
  - 3.3.1 Advanced Encryption Standard . . . 27
- 3.4 Asymmetric Cryptography . . . 27
  - 3.4.1 RSA . . . 27
  - 3.4.2 Digital Signatures Algorithm or Digital Signature Standard . . . 28

3.5 Hash Function 32
3.6 Practical Use of Cryptosystems 34
3.6.1 Public Key infrastructure 34
3.6.2 X.509 standard 34
3.6.3 Transport Layer Security 35
3.6.4 IP Security and Media Access Control Security . . . 36
3.6.5 Domain Name System Security Extensions 36
3.6.6 Border Gateway Protocol Security 37

III The third Part 39

4 The impact of Quantum Computer on cryptography 41
4.1 Quantum computer history 41
4.2 Quantum mechanics 43
4.2.1 Quantum phenomenal 43
4.2.2 Basics quantum mathematics and notation 44
4.2.3 Lov Grover algorithm 47
4.2.4 Peter Shor′s algorithm 48
4.3 Quantum resistant cryptography 48
4.3.1 Hashed-Based Digital signatures 50
4.3.2 SPHINCS+ 53
4.4 Adding Post-Quantum Signatures to Bitcoin 56
4.4.1 Cryptographic Keys and Addresses on Bitcoin 56
4.4.2 Base58 and Base58Check Encoding 57
4.4.3 Bitcoin Transaction and Validation 58
4.4.4 Proposal for Post-quantum Blockchain 60
4.4.5 Reason for Not Replacing the ECDSA Digital Signature 68
4.5 Conclusion 70

List of Figures

2.1 Bitcoin block 9
2.2 Merkel Tree Bitcoin 10
2.3 Merkel Tree summarizing 11
2.4 Libra transaction life circle [54] 18
2.5 Hyperledger project [39] 19
2.6 Blockchain Regular fork [13] 20

3.1 Digital signatures concept [63] 29
3.2 Elliptic Curve secp256k1 used by Bitcoin [5] 32
3.3 TLS [34] 35
3.4 IPsec tunnel overview 36

4.1 The Block sphere [36, 79] 45
4.2 Hadamard gate superposition. 46
4.3 Grover's algorithm [86] 48
4.4 W-OTS sign and verification process [16] 51
4.5 FORS Tree and Public key [45] 54
4.6 $SPHINC^{+}$ Full architecture [27] 56
4.7 Base58Check Encoding [4] 58
4.8 Pay-to-Public-Key-Hash (P2PKH) combining scriptSig and scriptPubKey [4] 59
4.9 Traditional Bitcoin address (left) and PQ Bitcoin address (right). 61
4.10 Post-quantum table of different public and signatures keys [24] 62
4.11 Iteration public key 65

List of Tables

2.1 Block field description for Bitcoin 9

Part I

The First Part

Chapter 1

Introduction

1.1 Motivation and Research Questions

The advancements in digital communications are one of the technological foundations of our modern society. Billions of people use social media, video streaming, payment, and communication devices daily. The world needs a better digital system to protect confidentiality, integrity, authenticity, non-repudiation in data transmission and data storage. Around 2009, a cryptocurrency known as Bitcoin, based on a blockchain system, or distributed ledger technology (DLT), was born. Bitcoin was intended to give people ownership of their money and not depend on the established financial system. Blockchain systems evolved rapidly to much more than that, and these systems are now used for securing the transmission and storage of data to be immutable against tampering.

In recent years, however the development in quantum computers has moved rapidly, and it is now predicted that within 20 years, quantum computers will pose a threat to current public-key cryptosystems, which we use daily for protecting our communication and devices.

Have you ever wondered how the information about our identity, our licenses, our medical record, our money, and our education is stored, and whether it is safe from tampering? Can we trust anyone to keep all our information secure and safe from tampering? Of course, we habitually trust the government and financial institutions to keep all this information safe and secure. But what if any of those institutions became corrupt and decided to change our identity records? This is where blockchains and distributed ledger technology (DLT) can be useful to protect the integrity of data and transactions without having to trust specific institutions.

In this thesis, we want to focus on investigating blockchain systems and how to protect these systems from quantum-computer attacks. We arrive at three questions that we want to do research and studies on:

Question 1: How can blockchain technology or distributed ledger technology be implemented in traditional transaction systems to strengthen their security?

Question 2: What is the threat from quantum computing against current blockchain systems and distributed ledger technology?

Question 3: How can we mitigate the quantum threat to blockchain systems and distributed ledger technology by using digital signatures based on post-quantum cryptography?

1.2 Research Method

We started out reading literature papers, articles, and books that are related to the blockchain system and distributed ledger technology for a better understanding of these topics. Then we looked a little more in-depth on different blockchain algorithms and how they work. After understanding some parts of the blockchain system, we continued trying to understand the security mechanism behind some of the blockchain systems. We spent a lot of our time trying to understand the basics of cryptography and how it is used in Blockchain systems.

We started again reading literature papers and articles that are related to a quantum computing and the potential threats quantum computers can pose. We spent a lot of our time trying to understand the basics of quantum mathematics or quantum physics and quantum algorithms "Shor's and Grove's".

After gaining sufficient understanding of blockchain systems and quantum computers, we went on to investigate how post-quantum digital signatures can form part of a post-quantum blockchain. Some post-quantum signature schemes are quite complex, and we chose the hash-based post-quantum signature scheme for our proposal, because we understand it most out of all the post-quantum scheme. There are already proposals for post-quantum blockchain systems, such as IOTA [41] and QRL [81], and Corda r3 [80] describes how to make blockchain systems more quantum-safe.

Finally, we proposed a post-quantum signature scheme for the cryptocurrency Bitcoin. We thought first to replace the Bitcoin digital signature ECDSA with hash-based quantum signature SPHINCS+, after researching and considering we felt that changing the Bitcoin ECDSA signature would cause some problem for Bitcoin. Our next idea was to protect the ECDSA signature instead of changing it. We wanted to keep everything about Bitcoin unchanged, and at the same time, protect it against quantum attacks. We analyzed the vulnerabilities of our proposed schemes.

Our final proposal was to use the Wintersnitz OTS signature. The basic idea is to hash the Bitcoin ECDSA signature and use the hashes as our public key, since the hash functions are more secure than ECDSA signature.

Chapter 2

Blockchain

This chapter describes some of the fundamental properties, concepts, and terminologies related to blockchain systems in general and related to digital money in particular.

2.1 History of Digital Money

Money is a medium for exchange that is represented in various forms of payment with a value that everyone can trust. It has three main functions: (1) it represents a medium exchange of value, (2) it is a unit of account for pricing, and (3) it provides a store of value for saving [10]. If we look thousands of years back when there was no economic system or money in use, people were exchanging goods for goods, a bag of rice for a bag of bananas, a pound of meat for a pound of fish. Soon people discover that it takes more effort to plant and harvest rice than to pick bananas. Catch fish cost less than breeding cattle. So naturally, people started trading and demand fair value for the product. As we can see, the measurement of the value of the goods is not stable. The arrival of the financial and economic system came in to make trades more balanced for exchanging goods. One can argue that goods are only worth what another is willing to pay for it.

In our modern world, the economic system is more complex than it used to be. In the past, the Central bank printed out real money, and then the money went to the banks and the banks distributed to the people that need the loans. Now, most of the money is created by the banks when they provide loans to people which the banks report daily to the Central bank [29, 62].

For example; If we borrow 1 million from the bank, the bank will credit 1 million to our bank account; new money has been “created”. When we pay back 1 million to loans, this sum will somehow be “deleted” from the bank system. Is to prevent the bank system not to cause inflation to the economy and lead to the financial crisis that happened in 2007-2009.

Almost everyone has a bank account that likely connects to an electronic card system such as Visa, MasterCard, Bank Accept, and mobile payment. IMF (International Monetary Fund) reports studies around the world that cash is falling in use, and that electronic money is increasing steadily [51].

We are moving to a cashless economy where everything is electronic and digitalizes. European Central Bank(ECB) defined electronic money in 1998 [9]:

" Electronic money is broadly defined as an electronic store of monetary value on a technical device that may be widely used for making payments to undertakings other than the issuer without necessarily involving bank accounts in the transaction, but acting as a prepaid bearer instrument. "

In 1983 David Chaum published a paper entitled "Blind signatures for untraceable payment" [19] that introduced the idea of digital cash using cryptographic protocols to ensure the complete privacy of users who conduct online transactions. The bank, which is issuing digital cash, blinds the content of a message before it is signed so that the bank can not determine the owner. David Chaum founded the DigiCash company in 1990, and the company went bankrupt in 1998, sold to eCash technologies, and late sold to Infospace. The DigiCash software system or eCash was a good idea, but it entered the market in the wrong era. The technology was ahead of its time, and there was not much support from the banks or institutions. Deutsche bank from Germany and The Mark Twain Bank was supporting the eCash system, but the bank later stepped back from supporting the eCash system. The cause of failure may be that the Internet was not fully integrated; buying and selling online service did not have many individual users at that time, and the use-case of the digital system was limited.

In 1996 came E-gold, a digital gold currency founded by Douglas Jackson and Barry Downey, which could transfer the electronic value of gold to other E-gold accounts in micropayment transactions. E-gold micropayment could be as small as one-thousandth of a gram of gold value, which is equal to less than 1 cent US dollar, where the gold price is around 55 US dollars per gram in 2020. E-gold was the first non-credit card payment-service provider and the only electronic currency that has achieved mass adoption with millions of accounts in 2008 around the time when the US Government shut it down. The reason for shutting it down was that E-Gold operated as an unlicensed money transmitter business in violation of federal criminal law, where the E-gold account was used for illegal activities such as money laundry, criminal abuse, and fraudulent Ponzi schemes [91].

Early examples of digital money or digital currency had the common factor that a bank or a company owned them so that it could be quickly taken down by the government's regulation and law enforcement. The next phase in the evolution of digital cash is the decentralized network blockchain system. A blockchain system is not owned by a bank or government, but by the individuals on the network.

2.2 Blockchain Technology

In October 2008, a person or a group called Satoshi Nakamoto released a paper entitled "Bitcoin: A Peer-to-Peer Electronic Cash System "[65].

In January 2009, the first cryptocurrency system was released using blockchain technology that could prevent the double-spending attack in payment transactions. Satoshi Nakamoto‘s real identity is still not certain. There is evidence about a few selected people who could know how to create this complicated system. Their names are Dorian Satoshi Nakamoto, Hal Finney, Nick Szabo, Craig Wright, Wei Dai, and David Kleiman. Some of the names had released one or two papers that describe systems similar to the Bitcoin system network before Satoshi Nakamoto released the Bitcoin paper.

Hall Finney, a cryptography enthusiast with a bachelor in engineering, was the first to receive Bitcoin transactions from the Bitcoin system [14, 15]. Hal Finney was a developer at Pretty Good Privacy (PGP) cooperation with Phil Zimmerman, and his concept of "Reusable Proof-of-Work systems" [30] published in 2004 looks similar to the Proof-of-Work concept in Bitcoin.

Nick Szabo, a computer engineer, published a paper on "Bit Gold" [67] to describe how to make a network with dedicated computer power for solving cryptographic puzzles that create a Byzantine-fault-tolerant network. This architecture is similar to that of the Bitcoin architecture system network. In May 2015 The New York Times published an article [71] about the Bitcoin creator; *"Myself, Wei Dai, and Hal Finney were the only people I know of who liked the idea (or in Dai's case his related idea) enough to pursue it to any significant extent until Nakamoto"*.

Dorian Nakamoto, a computer engineer, worked as a systems engineer on classified defense projects, and for financial information services companies. He is a physics teacher at Cal Poly University in Pomona. Dorian lived a few blocks away from Hal Finney. What is the probability that someone who has the same name Nakamoto, lives near Hal Finney and works with security programming in defenses system?

Craig Steven Wright is an Australian computer scientist and businessman. He has publicly claimed to be the part of the team that created Bitcoin, and the identity behind the pseudonym Satoshi Nakamoto. Wright claimed to have a Ph.D. in computer science from Charles Sturt University in 2015, but the University told Forbes that it only awarded him two Master's degrees, not a doctorate.

Wei Dai is a computer scientist who is best known as the creator of the Crypto++ library and the author of the paper entitled "b-money" [22] published in 1998 that talks about anonymous distributed electronic cash systems. This paper was credited in Nakamoto‘s Bitcoin paper for writing about anonymous distributed electronic cash systems.

Dave Kleiman is a computer forensics expert, who wrote many books related to security. Kleiman's most notable work took place at S-doc(Sealed Document Word) with the role of Chief Information Security Officer, where he developed a Windows encryption tool for NSA, NIST, and Microsoft Common Criteria Guidelines.

The person that most likely Satoshi Nakamoto is a group of enthusiasts that came together with an idea of creating a new and secure way for online payment, data transaction, and permanent data storage.

John McAfee founder of McAfee antivirus states that he knows whom

Satoshi Nakamoto is by giving clues; the answer lies in Satoshi Nakamoto white-paper [35]:

" If you read the whitepaper, number one, it's totally clear he's English — every single word that has a different spelling in English and American is English,"

"Number two, he left tells. There is only five percent of the population that has two spaces after a period — everything has two spaces after a period. And, the format of the document was identical to documents that he had published professionally,"

2.2.1 What is a blockchain?

A Blockchain is a linked data structure maintained by a decentralized Peer-to-Peer (P2P) distributed network of computer nodes, which uses cryptographic techniques to protect the integrity of data. Blockchain is based on a public digital ledger that is simultaneously maintained by a network of nodes that share the same data. This technology allows the system to create a trustless communication data systems, where transactions of data can be executed with transparency and reliability, without the need for intermediaries or any third-party interference. Everyone on the blockchain network has a record of all transactions that have happened in the past. In this way, we can check if any data records have been modified or tampered by attackers.

In principle, a blockchain is a decentralized database network that consists of blocks of cryptographically chained data. Each block in the blockchain is identified by a hash, generated by a cryptographic hash algorithm using SHA-2 (SHA-256, SHA-512). Each block contains a cryptographic hash of the previous hash that binds the new and old blocks. The first block in the blockchain is called the Genesis block and is block number "0".

2.2.2 Blockchain Block

A block is a collection of data containing records of metadata, records of transactions, and a block header hash. The block structure depends on the blockchain use-cases and the complexity of the consensus. A block in Bitcoin has 1MB block size and block time average every 10 min. Ethereum has a 2 KB block size, and block time average every 14 seconds, and Bitcoin Cash has up to 8MB block size. Let us look at an example of cryptocurrency Bitcoin block data.

Block field	Description
Version	Block version information
Previous block hash	The digital Hash value of the previous block
Merkle Root	The reference to a Merkle collection, which contains a hash of all transactions related to the block.
Timestamp	A time recording stamp, when this block was created.
Difficulty Target(bits)	The calculated level of difficulty target being used.
Nonce	A random number used to sign correct hashes by miners.
Magic Number	Identifiers what kind of file, data structure protocol it uses. bitcoins use 0xD9B4BEF99.
Block size	How large the block is. Each block in Bitcoin has a fixed 1 MB size.
Transaction counter	Counts of transaction in the block.
Transaction List	Stores all the transactions in the block.

Table 2.1: Block field description for Bitcoin

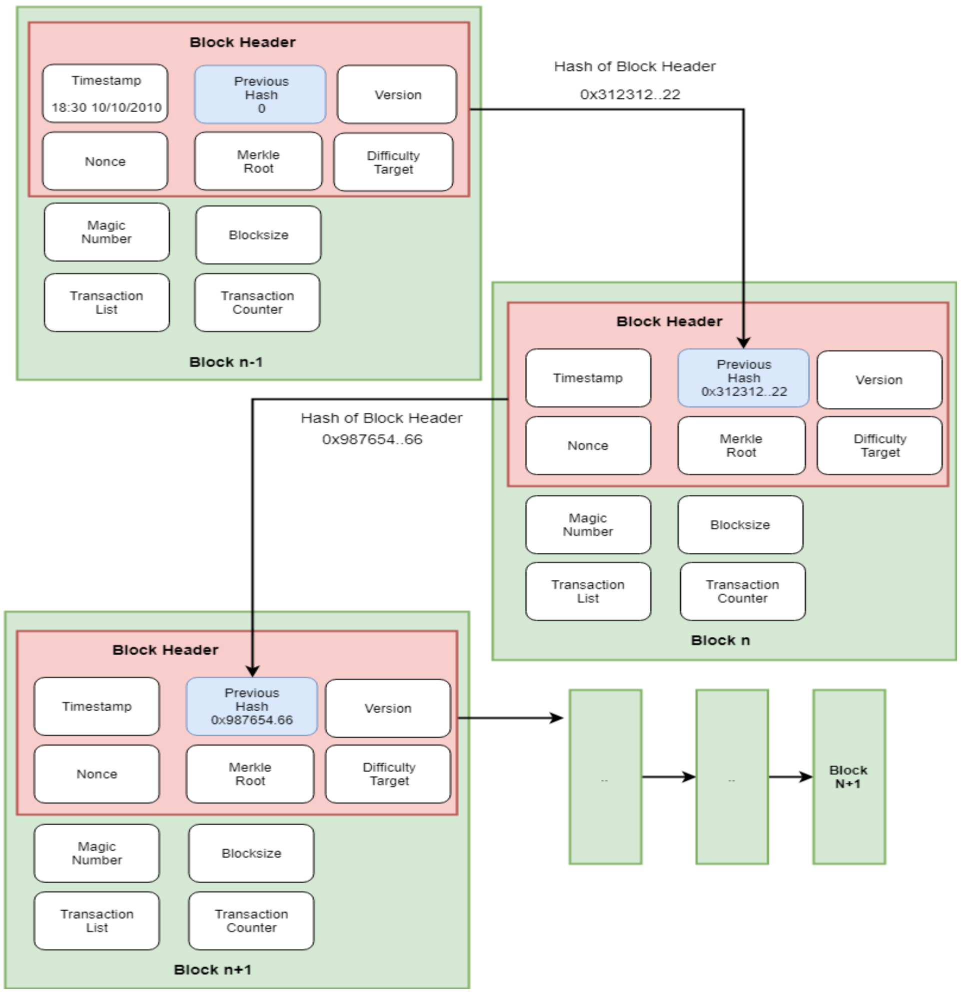


Figure 2.1: Bitcoin block

The previous block hash of every block header is inherited from the previous block, for every block n. For a block to be part of the blockchain, it needs to be given a valid hash. The block is stored in a Merkle Tree data structure.

2.2.3 Merkle Tree

Merkel Tree or Binary Hash Tree is a fundamental component in blockchain for structuring the ledger, it stored all data block as a leaf or a root node, and created an authentication path or Merkle path for verification of all transactions. The Merkle tree "Method of providing digital signatures" was patented by Ralph Merkle in 1979 [60]". A Merkle tree is a binary tree of hashes, where the root node has two children, and the children become parent nodes when they each have two children nodes. Using the Merkle tree provides integrity and validity, it allows a large data structure to converge efficiently, enabling secure verification using a small amount of data and make it easy to verify a transaction on the blockchain.

The perfect Merkle binary tree has the following properties: The number of leaves is always $2^n(n = 0, 1, 2)$.

- Each node has 0 or 2 children.
- All leaves are on the same level

The Merkle Tree structure diagram in Fig 2.2 below describes a hashing of the data text file using a binary tree, where data represents a transaction and is hashed. The binary tree does not branch before it is hashed. The data is hashed through a hashing algorithm, then the hashes of the two children are added together to create a new hashing root node. Then the root node becomes a child when it is combined with another hashing root node, as illustrated in Fig 2.3.

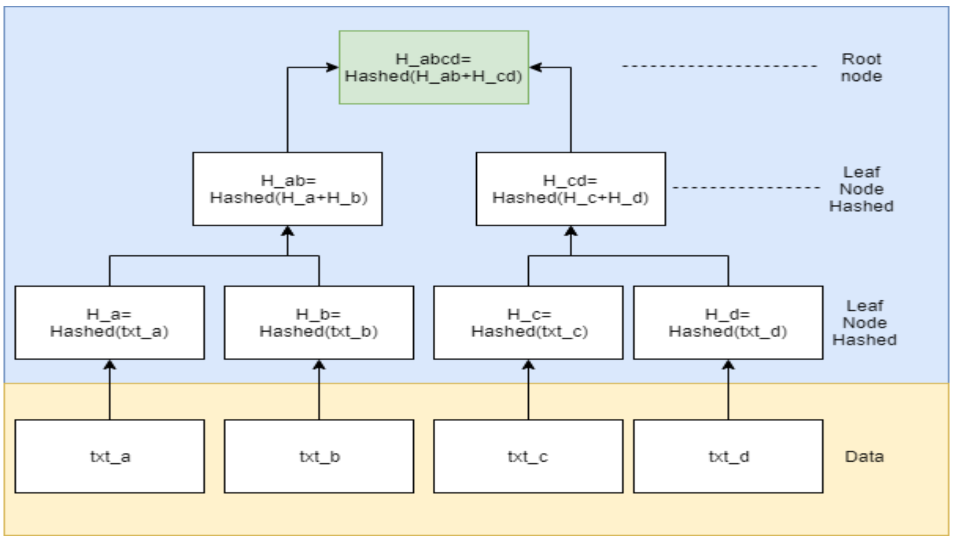


Figure 2.2: Merkel Tree Bitcoin

Assume that the hashes of the transaction a,b,c,d,e,f,g, and h are computed in the same way that bitcoin hashes its transaction using SHA-256. The leaf-node hash of the text_a transaction is: H_a = SHA256(SHA256(txt_a) and will have 32-bytes size.

- H_ab = SHA256(SHA256(H_a+ H_b))

Then hash the hashed H_a and H_b together, which in turn will create a new leaf-node of hash that has size 32-byte since hashes always have a fixed length. There is an average of 2000 transaction per block in Bitcoin [74]

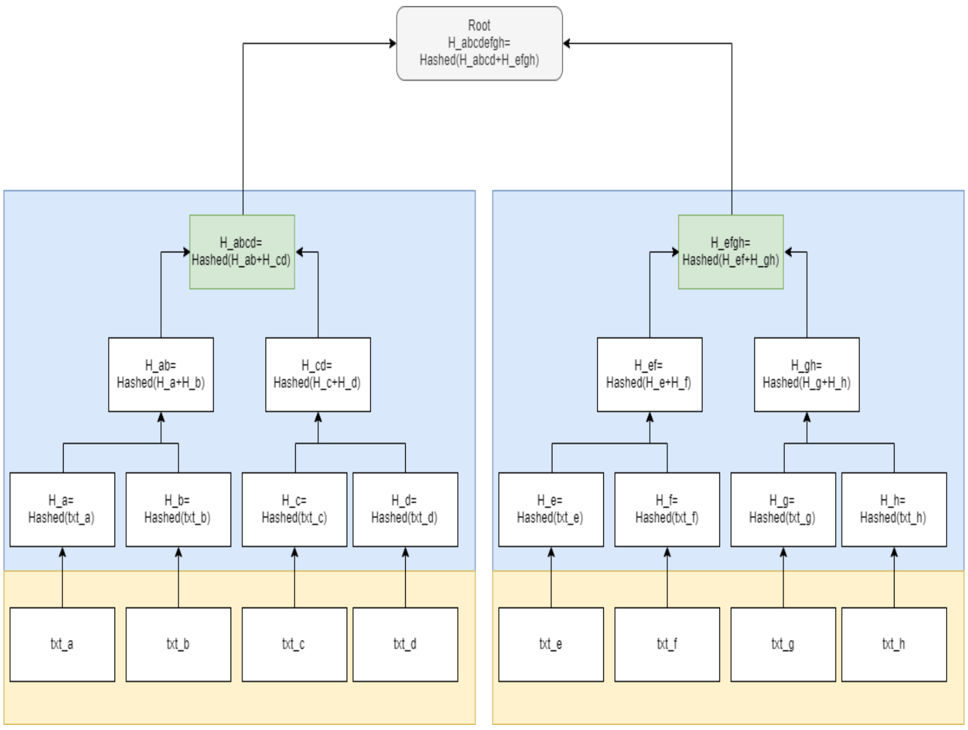


Figure 2.3: Merkel Tree summarizing

2.2.4 Mining

Mining is an essential part of the blockchain consensus and is a process to add new coins to the blockchain supply. New coins are the reward to the workers that participate in the blockchain mining activity network and transactions.

Permissionless Blockchains or public blockchains such as Bitcoin and Ethereum use Nakamoto's consensus algorithm Proof-of-Work (PoW) [65]. The PoW algorithm requires computer power to solve mathematical problems called the crypto puzzles or a random Hash to solve. The first node to solve the crypto puzzle gets the next coin reward. The difficulty of solving the crypto puzzle keeps everything in order and protects the network against fraud. The nodes or the worker with the greatest processing capacity has the highest probability of mining the next coin reward and with a little bit of luck.

A worker in the blockchain is called a miner or a forger. It is a node in the networks that work on collecting transactions, organizes them into blocks from the memory pool, and gathers them into a new block before starting the mining process.

2.2.5 Blockchain Consensus

In the blockchain system, a new data block is sent to the network for broadcasting, the node in the network has an option to accept or ignore the data block replica sent from the network. The more that accept the same replica and join in, the safer the network will become from dishonest nodes and tampering. The consensus jobs are ensuring that the network nodes reached a common agreement and understanding between nodes in the whole network. Reaching a consensus with a network which contains over thousands of compute nodes is not an easy feat for any system. There will be dishonest nodes, and some will fail to connect to the network. This is a known problem called Byzantine fault generals problem, and the ability to solve this problem is called Byzantine fault tolerance (BFT) [53]. The BFT is a type of system that can continue operating even when some nodes on the networks fail to work or are being attacked by a malicious actor. Therefore as long as more than half of the network nodes are reliable and honest, the BFT network will never go down or become vulnerable to any attack.

Blockchain is a BFT system architecture that uses a consensus algorithm based on Proof-of-Work, Proof-of-Stake, Delegation-Proof-of-Stake, etc. A good consensus algorithm ensures a stable transaction state on the network and an appropriate time for the nodes to reach an agreement.

2.2.6 Proof-of-Work

The Proof-of-Work is a consensus protocol used in many blockchain systems created by Satoshi Nakamoto. This algorithm originated from the Hashcash algorithm invented by Adam Back in 1997 [7]. The Hashcash algorithm uses a hash function as a building block, and it was used against Denial-of-service (DOS) and other attacks on general network security [8].

The Proof-of-Work is most known to be used in the Bitcoin cryptocurrency for its network protection against tampering on the Bitcoin ledger. The way the Proof-of-Work protects against tampering on its Bitcoin ledger is by using the cryptographic hash function to hash the transaction block. If there were any small changes to the original data, the verification would result in an unrecognizable hash, and the consensus will be denied this change. Generating valid hash blocks require computing power to solve a certain level of difficulty known as the crypto puzzle. To solve the crypto puzzle, the miners must find a random number in the range between 0 to 256-bits number that was created from the difficulty target of the hash, called a nonce. To find this nonce, we need to let the miner generate a random number and compare it with the nonce value. The miner will keep comparing the value until a match has been found. If some other miner on the network finds it first, then all other nodes must stop mining on that block and start with the next block. The miner that first finds the nonce gets the privilege to sign and receive block rewards. Miners use a digital signature algorithm based on the elliptic curve to sign and verify the transactions. Before a new block is created, the miner node adds a transaction block-reward, this transaction creates a new coin out of nothing and is the first transactions to be recorded in a new block. Every transaction on the blocks is hashed and put on a binary hash tree called Merkel Tree

2.2.7 Proof-of-Stake

Proof-of-Stake (PoS) was introduced in 2011 as an alternative to the Proof-of-Work (PoW) consensus for avoiding the enormous electricity consumption of the PoW algorithm. PoS and PoW share the same vision on how data should be maintained and protected, but the consensus process is different.

Peercoin the first cryptocurrency to introduce Proof-of-Stake was launched in 2012 [69], the PoS algorithm uses a combination of Randomization block selection, coin age-based selection, and the wealth of the nodes. The wealth of nodes means how many coins a miner or forger called in PoS, owns of the coin in their wallet. The forger locks a certain amount of coins in the blockchain network as their stake where the owner of the wallet sends a proportion of their coins to the blockchain network for earning network fees or transaction fees as a reward, instead of using computer power in Proof-of-Work algorithm to verify the block transaction and earn block rewards.

In a PoS blockchain, there are many different ways the network consensus lets the validator sign the block and get network fees. A coin such as Peercoin uses a combination of Coin age-based selection, randomization block selection, and owners Wealth.

Coin age-based selection methods are based on how long the coins are unspent and held for at least 30 days in the wallet, after 30 days the coin can start competing for the next block, and the coins can be held for a maximum of 90 days. Once a coin has been used in forging a block, its coin-age is reset to zero and has to start the count again. Randomization block selection means that the forger is selected by searching for the nodes with the lowest hash value and the highest stake, and the probabilities of being selected to validate a new block increases as a function of how long the coin is held in the wallet [70]. The combination of using the randomization and coin-aged will help so that the wealthiest nodes do not get to validate all the blocks.

The current crypto-currencies that use PoS at this point of writing are, e.g., Cardano coin [17], Tron coin[66], and Tezos coin [82]. These coins are ranked within the top 30 most valuable cryptocurrencies, and their values are estimated at over 500 million dollars on coinmarketcap.com.

2.2.8 Smart Contract

A smart contract is an exchange program code inside a blockchain system, where its job is to maintain a fair exchange of value assets automatically, without any interference. The system validates and automatically determines if the assets should be accepted or not by looking through the blockchain ledger. It works somehow like a real bank system, where we have a bank account and want to transfer money to another person. The bank system will let us transfer the amount of money we have inside our bank account to whomever we wish. If we have a zero bank balance on our account, we should not be able to send money to anyone.

The smart contract has many use-cases, depends on what blockchain system solution one chooses to use. Some blockchain system has a smart contract as a digital contract where two or more people follow the contract rules, when the contract condition is met, the smart contracts automatically write to the blockchain ledger and will be stored away for later use as proof.

2.3 Permissionless Blockchain

Since the introduction of the Bitcoin cryptocurrency in 2008-2009, there are now more than a thousand different cryptocurrencies [21]. There are two main categories of blockchains, one is Permissionless blockchain or public blockchain, and the other is Permissioned blockchain.

A Permissionless blockchain, such as Bitcoin and Ethereum is a decentralized system that anyone can be a part of. Anyone can write to the block through transactions, and anyone can participate in the consensus process of mining, voting, and protecting the network. The Permissionless blockchain has, by nature, a transparent transaction ledger that lets all the nodes in the network access all the information. However, there are also blockchains such as Monero, ZCash, etc. that focus on hiding transactions by obfuscating the transaction data so that no one can understand or find where transactions come from.

2.3.1 Bitcoin

Bitcoin is the first digital currency or cryptocurrency to manage currency transactions online without a centralized trusted third party. The Bitcoin whitepaper was published in 2008 by Satoshi Nakamoto [65]. The Bitcoin genesis block was operational on 23 January 2009, and 50 bitcoins were mined. Bitcoin's primary creates a ledger that connects computer nodes and shares the data file simultaneously to all the nodes on the network. Anyone can be a part of the Bitcoin network by installing the Bitcoin wallet ledger [12] to their devices, such as a computer, mobile phone, or tablet. The wallet should stay online for updates and help the network distributed data. With the full wallet node or Full Client, we can participate in the Bitcoin mining process.

Bitcoin mining requires a tremendous amount of hardware power to find the nonce. The computing power is measured in terms of hash rate, which is the number of attempts per second on guessing the nonce value. A specialized miner equipment hardware AntMiner S9 is capable of 12,930 GH/s, currently Antminer s9 consumes around 2,663,863 kWh electricity for 75 BTC, which is around 35,518 kWh per Bitcoin. Assuming a typical electricity price of NOK (Norwegian kroner) 0.5 per kWh, the cost of mining a single Bitcoin is around NOK 15,000. Miners with relatively low computer power will have to cooperate in a federated mining pool to solve the puzzle. This will increase their chance of finding the nonce and get the block reward. Bitcoin block rewards will be halved every 210,000 blocks, the next halving will be in around 2024-2025, and the payout for rewards will be 3.125 Bitcoin per block, instead of 6.25 Bitcoin per block. The network creates a new block every 10 minutes, the block size is 1 MB, and the total supply of Bitcoin is limited to 21 million coins. When Bitcoin reached its total coin supply, the miners of Bitcoin will be paid in forms of transaction fees as coin-rewards.

Bitcoin payment transactions are transferred between a pair of entities sending digital money currency to each other without any central authority such as banks. Without the bank as the middleman and no third-party interference, the transaction of digital money can potentially be much more effective and faster. An ordinary bank's transfer system takes at least 12 hours for a transaction between two countries to happens. Furthermore, a bank transaction always requires information about the merchant that sells the product and the cardholder or bank account that wants to buy the product. With Bitcoin transactions, there is no need to provide any personal identification, and it just needs to show the Bitcoin public key address. The Bitcoin does not depend on any physical form and exists only virtually. The only way to generate Bitcoins is by mining it with computer power.

Below is a practical example of how the Bitcoin cryptocurrency is used in real life:

Alice heard about Bitcoin on the news and from her friends talking about it. She wants to learn and how to use it. Alice finds an excellent manual on how to get started with Bitcoin:

- Alice has to visit Bitcoin's main homepage and download the real wallet in https://bitcoin.org/en/download. Beware that there are many fake Bitcoin homepages.

- Alice has to downloads the Bitcoin client software as an application or web application. The software exists in Full client or Light Client. The Bitcoin client is a wallet.

- The Full client stores the entire history of Bitcoin or full ledger, which contains over 200 GB of data [56]. If Alice downloads the full wallet, Alice will have full access to the wallet and can participate in the mining process.

- The light Client stores only the block header of the Bitcoin history, its use for validating the authenticity of transactions by using third-party servers for access to the Bitcoin network. Alice has to trust the third-party server if she chooses to download the light client.
- The third option is to use through third-parties exchange. This option is not so safe as it has been reported that third-party exchange are regularly at risk of being hacked. Examples of Decentralized exchanges such as Binance, Bittrex, Coinbase, etc. This option implies to use a Web Client through a web browser and store the user's wallet on a server owned by a third party. Here Alice can send, receive, selling, and buying the cryptocurrency.

Alice chooses to downloads the light client application on her mobile phone. The light client gives Alice a Bitcoin Address and a passphrase. The Bitcoin address is Alice's public address or Alice's Bitcoin account that she can share with anyone she wants. The passphrase is the private key that Alice has to keep secret from others. It is recommended that Alice changes the passphrase to her passphrase, print out the passphrase on paper, and put it on an external storage device. If she loses the passphrase, she will lose access to her Bitcoin address, and no one can help her retrieve it.

Alice wants to buy Bitcoin to her Bitcoin address, but there are no banks that support a direct transaction between the bank money and the cryptocurrency. at this time writing. Alice can choose some options below.

1. Alice can ask a friend of hers that she knows to send her Bitcoin, and she will pay real money in return.
2. Alice could find a seller on the online website, "localbitcoins.com". This is not recommended since the seller could take the money and never send her the Bitcoin.
3. Alice can buy from an exchange that specializes in buying and selling cryptocurrency such as Coinbase, Coinmama, etc.
4. Alice can try a nearby located Bitcoin ATM, which is a platform for exchange from real money to Bitcoin.

Assume that Alice wants to try sending 1 Bitcoin to her friend Bob. First, she needs Bob's Bitcoin Address. Alice uses the light version Bitcoin wallet app, scan the QR code that contains Bob's Bitcoin address or writes Bob's Bitcoin address. She sends 1 bitcoin to Bob's Bitcoin address. Bob checks his wallet app after 30 min for Alice's 1 Bitcoin transaction. Alice will receive a confirmation of the block transaction, which shows that the transfer is complete.

2.4 Permissioned blockchain

A Permissioned blockchain is also known as a private blockchain. It is operated by a known entity where the blockchain is owned by a company, enterprise or organization, that controls the network's restrictions on who can be delegated in the network and the block transactions. The main difference between the Permissioned blockchain and the private blockchain is that the private blockchain is not open to participating in the network for anyone, where the Permissioned blockchain has some criteria such as "Know Your Customer (KYC)" for the public to join. KYC is a bank regulation term, a process to verifying the identity of the person for ensuring that money laundry activity does not occur.

In a permissioned blockchain, such as Hyperledger Fabric [39] and IPFS [44] every entity node must be known to the network, the nodes must be verified and identified by a trusted certificate. The permissioned blockchain is more suitable for a centralized system authority than the Permissionless blockchain.

2.4.1 The Interplanetary File System (IPFS)

The InterPlanetary File system (IPFS) [44] is a peer-to-peer distributed storage network that seeks to share information by storing information on devices, where devices can be shared with other devices through the network of IPFS. IPFS uses content addresses for sharing the information instead of a location-based address. The content is stored in fragments of data with different sizes and is spread through the network with a unique identifier hash that will recollect all the data in that files to become a complete content file. When a file is missing a part of its data, the system reconstructs it by looking for the older patch in the filesystem tree.

The IPFS content address is structured, for example [43]:

- /ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Aardvark.html

Where in comparison a traditional URL; for example:

- https://en.wikipedia.org/wiki/Aardvark
- /Users/Alice/Documents/term_paper.doc

The IPFS [42] system is based on a Peer-to-Peer system such as BitTorrent, Github Version Control Systems, Distributed Hash Tables (DHT), and Self-Certified Filesystem (SFS). BitTorrent is a peer-to-peer file-sharing system, which coordinates a network of untrusting nodes to cooperate in distributing files to each other. Distributed Hash Tables are used to coordinate and maintain metadata in the peer-to-peer system using Kademlia [59]. Self-certified filesystems let the user verify the public key offered by the server, negotiate a shared secret, and secure all traffic. Version Control Systems Git provides a powerful Merkle-DAG object model that captures changes to a filesystem tree. Merkle Dag tree is constructed from a Merkle tree with a Directed Acyclic Graph (DAG), which has the property not to rewrite the node that has already been encountered. Merkle Dag allows multi chains of blocks to exist in parallel and interconnected, while never changing their parent node.

2.4.2 Libra Blockchain

Libra blockchain or cryptocurrency LBR is a decentralized blockchain proposed by Facebook in 2019. It is a permissioned blockchain which is planned to support a low-volatility cryptocurrency solution and a smart contract platform that together aims to enable a simple global currency and financial infrastructure that could be adapted by millions of people [55].

A distributed network of validator nodes maintains the Libra Blockchain. The validators collectively agree on an ordering of transactions in the blockchain. The transaction fees call gas price, is a fixed point in time. The leader of the block gets to validate a block of transactions. There is no place in the white-paper [55] saying which node can become the leader and who will get the transaction fees.

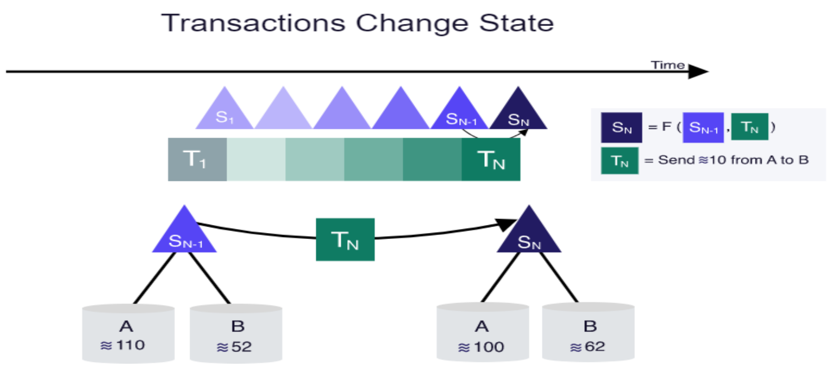

Transactions Change State
Time
S_1
S_{N-1}
S_N
T_1
T_N
S_N = F (S_{N-1} , T_N)
T_N = Send ≈10 from A to B
A ≈110
B ≈52
A ≈100
B ≈62

Figure 2.4: Libra transaction life circle [54]

Libra Transaction definition:

- S stands for a Sequence number.
- T stands for Raw transactions.
- A is Alice and B is Bob.

2.4.3 Hyperledger

Hyperledger is an open-source global collaborative blockchain technology. it is hosted by The Linux Foundation, including leaders in finance, banking, Internet of Things, supply chains, manufacturing, and Technology.

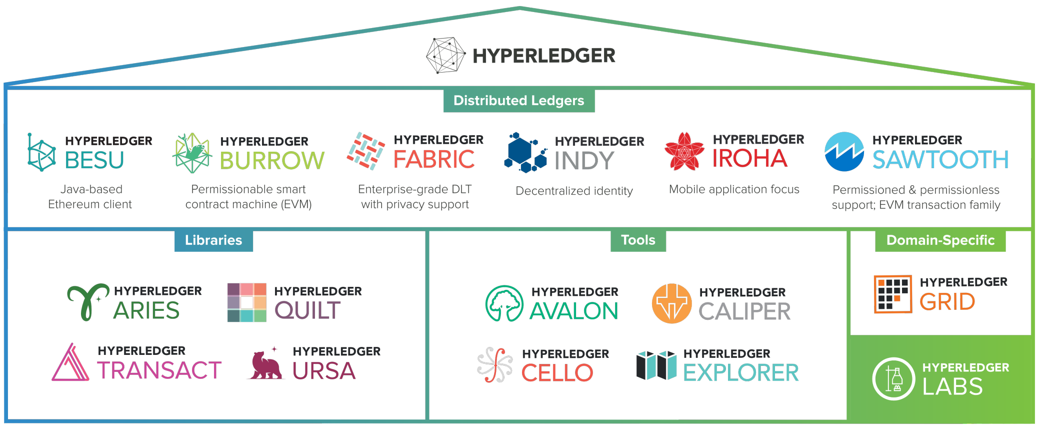


Figure 2.5: Hyperledger project [39]

Hyperledger Fabric is one of the Hyperledger projects that are more known to others with a collaboration with IBM at the start. Now there are around 35 organizations in the project [37]. Hyperledger Fabric is a permissioned distributed ledger framework for developing solutions and applications for industry use-cases for preserving privacy [38].

Hyperledger fabric uses a smart contract called chaincode for business and enterprise use-cases, where they want to create a safe and secure communication channel with payment between clients, where the information has to be kept confidential.

2.5 Fork on Blockchain

A fork is a duplicated of the original program, where the duplicated version has a new implementation and direction to diverge from the original program. In computer programming, a fork is when a developer copies an existing project and implements a new thing, so it becomes a new software based on the original code. In a system, a fork is a running process that a parent process created as a child process, which has inherited some part of the parent process.

In a blockchain, there are three ways a fork can occur one is Hard fork, the second is a soft fork, and the third is a regular fork when two or more miners solve the puzzle simultaneously and attain the same answer but have a different hash number.

In cryptocurrency such as Bitcoin, the case of regular fork occurs when there are identical blocks for the miners that have to solve the puzzles. The winner miners will have to wait for the system to decide on the winner of the block, but at that time, all other miners will compete for the next block. Since there is more than one block, the miners will have to mine a different block depending on what block they receive.

For example; If one lives in Brazil and the others one in Japan, and they found the answer simultaneously. They broadcast their answer to the network; some will receive the answer from Brazil first, and some from Japan first. After around 6 blocks, the system will decide the winners of the block that has the longest chain and abandon all other identical blocks. The abandoned blocks will become an orphaned fork that will be included in the ledger, and all the nodes in the network will have the same ledger once again.

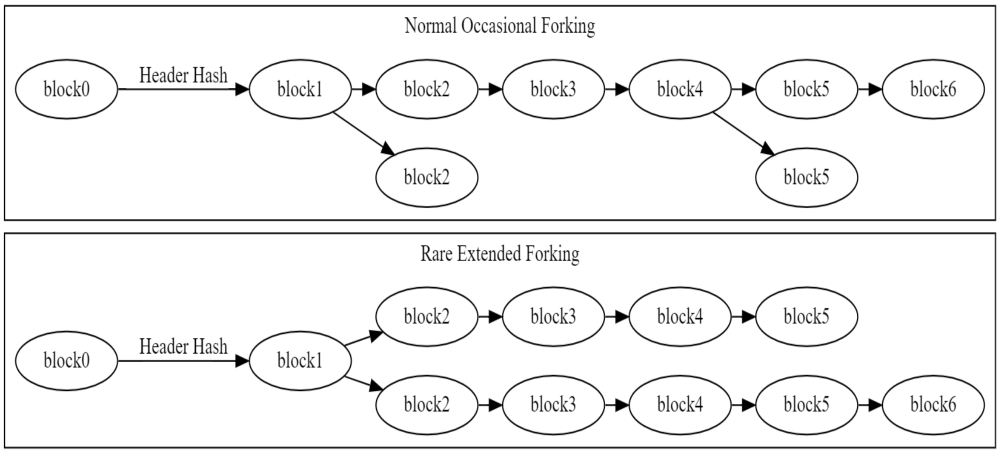


Figure 2.6: Blockchain Regular fork [13]

A hard fork or a soft fork is intentionally changing the consensus rules of the system software by implementing new consensus rules suggested by the developer and miners vote. Permissionless blockchains such as Bitcoin and Ethereum do not have any central authority that holds power to decide what can be changed on the blockchain. Making a change on the consensus and protocol needs miners vote, the developer of the coin can only suggest a proposal of how the coin can be changed to new features and methods to prevent attacks.

A hard fork occurs when there is an argument between miners on how they want the coin consensus rules to be changed. The miners vote to decide on the change on the coin, where the miners that support in the new consensus rules will have to accept and upgrade their nodes. The miners that reject the new consensus rules do not do anything, they will continue mining according to the old consensus rule.

For example; Bitcoin hard-forked to Bitcoin Cash, where Bitcoin Cash changed the Bitcoin block size from 1MB to 8MB, trying to solve one of the Bitcoin scalability problems. When a hard fork occurs, a new coin is created, and the holder of the old coin will also have the right to have the new coin in the wallet. Soft-fork occurs where the developer and miners abandon the old consensus rule and support the new consensus rule. No new coin will be created in this case.

2.6 Attacks on Blockchains

Security in the blockchain is critical for the integrity of transactions, availability of the system and consistency of the nodes across the network, and the prevention of double-spending.

Blockchain security design and implementation emerge from using a Merkle-Tree data structure to combine multiple documents into one block, where each block ensures the integrity of the previous block. Each transaction is hashed with the SHA-2 family of hash function algorithms. The data block is signed by a validator with a digital signature algorithm elliptic curve (ECDSA) that prevents tampering on the transactions.

One of the main hidden components in a blockchain system is the use of the Peer-to-Peer (P2P) network connections between nodes for automatically sharing of data. A Peer-to-Peer (P2P) network consists of computer nodes that connect via the Internet by sharing data files with each other. Data files can be shared directly between systems on the network. An attack on a P2P network is easier and less expensive than attacking the whole blockchain system. Since the blockchain consensus needs to propagate the updated block ledger to all the nodes in the network, an attack could happen before all the nodes are updated.

Eclipse Attack

An eclipse attack means attacking a specific part of the network nodes, isolating the victim nodes incoming and outgoing connections by surrounding the victim's node with dishonest nodes [6]. Isolating nodes from the network can make the network more vulnerable if the attacker chooses the most significant contributing nodes on the network. The network will become weaker, with better chances for the attackers to modify the ledger and to follow up with other attacks. The attackers could also isolate the victims from the network, and sends the victims a false ledger, trick the victims into believing it is the real ledger and try to steal from the victims.

Sybil Attack

A Sybil attack is an attack by creating several identity nodes on the same network, where a single entity controls all of them. The attackers are trying to have more influence on the networks [1, 26]. If the attacker controls a considerable part of the nodes on the network, the attacker could try to tamper with the ledger, change the consensus rules, and get all the block rewards.

51% Attack

A 51% attack is an attack to take control of over 50% of the network hashing power. Control over the majority of the network attacker can interrupt other miners from recording new blocks and allows the attacker to perform a double-spending attack on the network.

A double-spending is an attack where the same transactions are spent more than once. The attacker could use any attack as long it has a higher proportion of the network computing power such as Eclipse attack, Sybil attack, 51%, etc. When an attacker has control over more than 50% of the network, it can modify the transaction on the block or any data [46].

A double-spending attack requires a large amount of computational power and resources, and it also depends on the size of the network nodes. Cryptocurrencies such as Bitcoin has over 10 thousand full network nodes operating, which makes it nearly impossible to use any kind of attack on the network. Some cryptocurrencies have been victims of the double-spending attack, such as Bitcoin Gold [31]. A cryptocurrency that has a small network of nodes will be an easy target for double-spending attacks or any other network attack.

Part II

The Second part

Chapter 3

Cryptosystems

3.1 Introducing to Cryptography

Cryptography is a security mechanism for protecting our data integrity and privacy in a modern digital infrastructure world, where everything is soon to connect to all devices. We bring it with us a communication device, such as a mobile phone everywhere we go. We take pictures, write notes, surf, watch movies on the devices. We possess several devices that connect to the wireless network WiFi or Bluetooth at work or home [18], to protect our device's data privacy and integrity from eavesdrop, unauthorized access, or other malicious actions on our devices, we implement in cryptography mechanism. The cryptography mechanism can not stop all the attacks; it can however make an attacker give up by encrypting the data with mathematics hard problems that can take years to solve.

3.2 Security Concepts

Confidentiality

Confidentiality refers to information that can only be accessed by authorized parties. Access to information should only be disclosed to those who are authorized to view the required data. Failure on confidentiality will leads to data being revealed.

For example; A hacker has managed to get access to a secret government database and can read top-secret information stored in the database. The attacker could choose to reveal information to the public; when it is revealed, it can not be undone. Reveal information could damage the reputation for the government, or they could try to blackmail with ransomware encrypt the database system with cryptography.

Maintaining the confidentiality of the data is a critical security goal for all systems. It can be challenging to protect the confidentiality of data when it comes to an eavesdropper, bribery, brute-force attack, or sending the information to the wrong individuals.

Integrity

In information security, data integrity means that unauthorized entities have not tampered the information. A secure computer system with integrity should have the property to stop tampering and hard to breaching the integrity, and if it happens, it should be detectable and recoverable [73]. Imagine if someone could modify or tampered the data on our bank account, money in that bank account will be gone and used.

Availability

In information security availability of systems is important because we expect the system to operate and functioning 100% when it is needed. For ensuring the availability of systems, it needs to be a fault-tolerant system, in case of power outages, hardware failures system, and network overrun, addition to updates of the system and attack from malicious actors.

Fault tolerance means that the system should have the properties of reliability, safety, maintainability, and availability. The most common attack on availability is by attacking the system network server using Denial of Service (DOS) to force the network to slow down or to shut down due to overload on service.

Accountability

In information, security accountability is directed to the users that use the system to make sure that all actions are being logged and can be traced back to users.

For example; Assume that users with admin privilege visit a risky infected homepage and causes the company to be attacked. The system should have the ability to know who caused this incident and let that individuals take responsibility.

Non-Repudiation

Generally, non-repudiation means that there exist undeniable evidence on what an individual has done. Non-repudiation means one cannot reject or deny having sent or received a message or executed an action. The evidence will be in the form of digital data, such as digital signatures or some kind of authentication data as video, picture, message, or log. With the evidence, we can prove that in the time of the action occurs, and there is a data-trace that belongs to the individuals that have committed these actions.

3.3 Symmetric Cryptography

A symmetric-key algorithm is a cryptographic system where the same key is used for encryption and decryption. Symmetric cryptography is using on the Internet for securing the transmission in end-to-end data transport.

3.3.1 Advanced Encryption Standard

Advanced Encryption Standard (AES) was published in December 2001 by NIST [20, 75], after having been select as the winner out of 15 candidates. AES replaced the Data Encryption Standard (DES) after over 20 years in service. DES is vulnerable against cryptanalysis and brute-force attacks. The Electronic Frontier Foundation (EFF) proved that the DES could be broken in 1998 [32].

An AES has a fixed block size of 128 bits and has the key lengths 128, 192, and 256 bits. 10 round for 128 key bits, 12 rounds for 192 key bits, and 14 rounds for 256 key bits. Each round has an operation of SubBytes, ShiftRows, MixColumns, and Addroundkey. SubBytes substitutes every single byte by another value. ShiftRows permutes the Bytes in each row, MixColumn performs a linear transformation on each column of the matrix, AddRoundkey adds the round key to the whole matrix.

An AES-128 bits encryption starts with an initial round: AddRoundKey. In the Main round, it repeats 9 times: SubBytes, shift rows, MixColumns, and AddRoundKey. The last round will not have MixColumns: Subbytes, ShiftRows, and AddRoundKey.

3.4 Asymmetric Cryptography

Asymmetric cryptography or public-key cryptography (PKC) is a form of encryption and decryption for data, that uses a pair of keys; a public key and a private key. The public-key is shared between multiple entities and can broadcast to the public. The private key needs to be hidden in secret, and the one needs to know the private key is the owners of it. The private key is the proof of authentication to the public key; it used to decrypt data sends to the public key.

Asymmetric cryptography is a fundamental security building block in our modern cryptosystem, is used in numerous devices, applications, and software to ensure integrity, authenticity, and non-repudiation. The most used algorithms related to asymmetric cryptography are Elliptic-curve, RSA, and Diffie-Hellman.

3.4.1 RSA

RSA is an asymmetric cryptosystem algorithm widely used for securing the transmission of data on the Internet. The security of the RSA algorithm uses integer factorization hard problems. The key generation is based on the product of two large prime numbers; hence the two prime numbers must be kept secret. Knowing the two prime numbers, we can recreate the public key and private key. The public key is the encryption key using for encrypting the messages; this key can be shared with the public. The private key is the only key that can decrypt the encrypted data. RSA invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977.

```
Algorithm 1: RSA key generation [33, 61]
Output: Private key sk, public key pk
1. Choose two random large distinct primes $p$ and $q$ with $p > 2^{n^{-1/2}}$,
   $q < 2^{n/2}$. Where $p \neq$q .
2. $n$ is a product of $p$ and $q$: $p \cdot$q $\rightarrow$ n .
3. $L$ is the value of Euler's $\varphi$(n)$function$ :(p-1)(q-1)$\rightarrow$L.
4. Choose a random $e \in \mathbb{N}$ such that $1 < e < L$ and gcd $(e, L) = 1$. e
   is the exponent.
5. Calculate the inversed $d$ of $e$ in $\mathbb{Z}_L$. We are using the Euclidean
   algorithm. Where $d \in \mathbb{N}$ with $1 < d < L$ such that $ed \equiv 1$(mod L).
6. Publish the public key $pk$= $(n, e)$ and keep the secret key
   $sk$=$(n, d)$.
```

RSA encryption and decryption: Given plaintext messages m and ciphertext c [33, 61].

1. Alice obtains Bob public-key (n, e) from the database.

2. She encrypts the message (m) by computing to ciphertext $(c) \equiv m^e$ (mod n).

3. Alice sends messages c to bob.

4. Bob receives messages c, Bob uses his private key (d) to compute m $\equiv c^e$(mod n).

3.4.2 Digital Signatures Algorithm or Digital Signature Standard

A Traditional paper document contract is signed between two or more people for reaching an agreement, signing on the agreement is proof that the agreement has been reached. Get a copy of that agreement document is proof of the agreement has happened.

A digital signature is a signature we use to sign the digital document and used to verified many authenticated services on the computer. The digital signature is a data block that acts as verifying for proof of ownership ad authentication. Each digital signatures has its unique number of the pattern just like a human fingerprint, where all ten fingers are different, and all those are different from all other people living on this planet. Digital signatures are used as a standard for digital fingerprint, software distribution, software detection, financial transaction, and everything that needs to protect against tampering and identity theft.

A digital signature algorithm consists of a signing process and a verification process. A signed agreement uses the signing process to generate a digital signature on data. A verifier uses the verification process to verify the authenticity of the signature. Each signed agreement is associated with the public key and the private key, and they are the owner

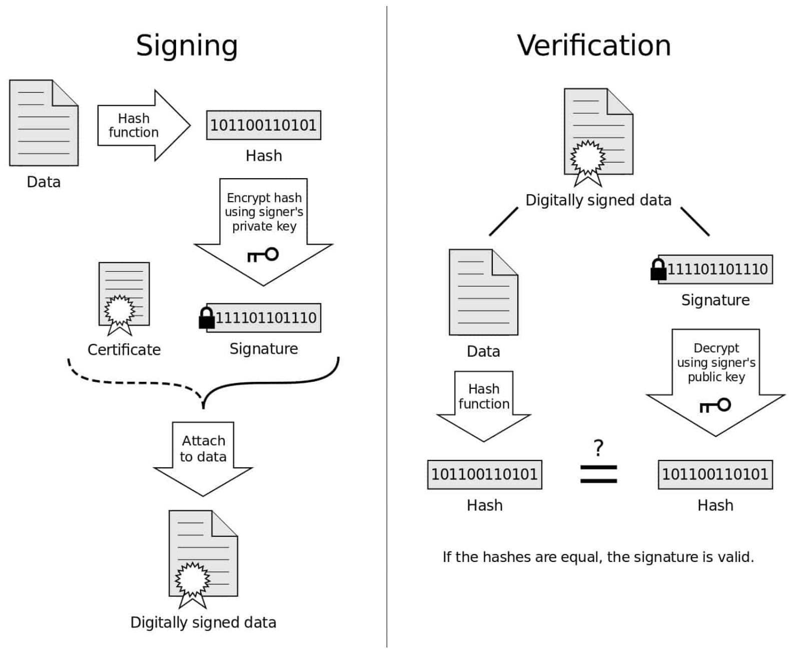


Figure 3.1: Digital signatures concept [63]

of that key pair. The owner of the key pair is the only person that is authorized to use the private key. To prevent other entities from declaring being the keys pair owner and using the private key to generate fake signatures. Hence, the private key must remain a secret.

The requirement of digital signatures are [33, 50]:

- The signature must be tightly attached to the signed document.
- It should be easy to sign for the legitimate signer, easy to verify the signature for the recipient, and at the same time hard to forge a signature.
- The signer should not be able to deny that he or she has signed the document.
- Sometimes it is crucial that a signed document can only be used once for its legitimate purpose, not several times.

The DSA Domain parameters are described below NIST [77]:

- x the private key that must remain secret; x is a randomly or pseudo-randomly generated integer, such that $0 < x < q, i.e, .x$ is in the range $[1, q-1]$.
- p is a prime modulus, where $2^{L-1} < p < 2^L$, and L is the bit length of p. Values of L can choose 1024-bits, 2048-bits, and 3072-bits. This value can go higher. NIST recommends over 1024-bits.

- q is a prime divisor of $(p-1)$, where $2^{N-1} < q < 2^N$, and N is the length of the bits q, Values for N can choose 160-bits, 224-bits and 256-bits. These values can go higher. NIST recommends over 160-bits.

- g a generator of a subgroup of order q in the multiplicative group of Galois Field, GF(p), such that $1 < g < p$.

- The public key y, where $y = g^x \bmod p$.

- A secret number k that is unique to each message; k is a randomly or pseudo-randomly generated integer, such that $0 < k < q, i.e., k$ is in the range $[1, q-1]$.

- k^{-1} is the multiplicative inverse of k with respect to multiplication modulo q; i.e., $0 < k^{-1} < q$ and $1 = (k^{-1}k) \bmod q$.

- The signature of a message M consists of the pair of numbers r and s, and will is written (r, s).

- r = (g^k mod p) mod q.

- z = the leftmost $min(N,\ outlen)$ bits of $Hash(M)$. $outlen$ is the bit length of the hash function output block.

- s = (k^{-1} (z + xr)) mod q.

Algorithm 2: DSA signature generation

```
System Parameters: Private key $x$, secret number $k$, Generator
  Point $g$, $p$ prime modulus, $q$ prime modulus, signature $(r,s)$, $h$
  hashed algorithm.
    1. Select $k \in_R [1, q-1]$.
    2. Compute $T = g^k \bmod p$.
    3. Compute $r = T \bmod q$. If $r = 0$ then go to step 1.
    4. compute $h = H(m)$.
    5. compute $s \equiv k^{-1}(h + xr) \pmod q$. If $s = 0$ then go to step 1.
    6. Return Signature $(r,s)$.
```

Algorithm 3: DSA signature verification

```
System Parameters: $p$ prime modulus, $q$ prime modulus, signature
  $(r, s)$, $h$ hashed algorithm, $M'$ $r'$ $s'$ receive of $M$ $r$ $s$, public key $y$,
  verification value $v$.
1. Verify that $0 < r' < q$ and $0 < s' < q$ are integers in the interval
  $[1, q-1]$ if any verification fails then return("reject the signature").
2. If the two conditions in step 1 are satisfied, the verifier computes
  the following:
3. $w = (s')^{-1} \bmod q$.
4. $z$ = the leftmost $min(N, outlen)$ bits of $Hash(M')$.
5. $u1 = (zw) \bmod q$.
6. $u2 = ((r')w) \bmod q$.
7. $v = ((g)u1(y)u2) \bmod p) \bmod q$.
8. If $v = r'$, then the signature is verified or else invalid.
```

Elliptic-curve cryptography

Elliptic curve cryptography (ECC) is asymmetric cryptography where the security based on the discrete logarithm problem of the hardness of the Elliptic Curve. The algorithm started at a random point in the elliptic curve than it adds an unknown random point on the prime field of an elliptic curve to the started point. Take the sum of the start point and next point than reflect the calculated sum point to the opposite of the curve, this will be our public key, and the started point will be our private key. The unknown random point is a hard problem that needs to be solved, and the bigger size elliptic curve is, the harder it is to find where is has jump and how many times.

ECC is used for digital signatures, pseudo-random generator, and key agreement with Diffie Hellman as the Suit B algorithm recommends by NIST. For now, there is no mathematical algorithm that solves effective enough to reverse-engineer the process to determine the starting point and the random point bounce of the elliptic curve. Our only solution is to use an exhaustive search for all the points and try subtracting or adding the number until we find the private key, which is considered computationally hard.

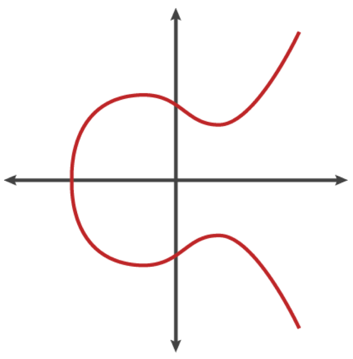


Figure 3.2: Elliptic Curve secp256k1 used by Bitcoin [5]

Elliptic curve key generation public key (pk) and private key(k):

1. Generated two random number points on the curve. Connect these two points with a straight line. The point is coordinated by (x, y). The point in x and y values can be huge numbers, depend on the curve size.

2. Multiply it with the generator point (G) on the curve. The generator point is a predetermined point that been selected. Multiply k with G.

3. The public key (pk) is a resulting of k multiplying with G, $pk = k * G$.

Cryptocurrency Bitcoin using a hash function algorithm SHA-256 and RIPEMD-160 with ECC, and Ethereum uses Keccak with ECC.

3.5 Hash Function

A hash function is designed to take an input string of any length and produce a fixed-length output value. Hash functions are often called oneway functions because of their property that it is easy to compute the hash and hard to reverse back to the original data. When the hash function converts data, the output becomes a unique pattern and deterministic in the sense that, for the same input, it produces the same output every time, if there is the smallest change in the input, the output will not have any similar pattern with the old output. Hash functions are vulnerable to the rainbow attack, where the rainbow table stores the hash and the original input of the hash number. We can run the unknown hash against a rainbow table and let it search for the original string.

Hash functions used in different security applications and system: indexing data in hash tables for lookup, for digital signatures such as fingerprint, for message authentication, for check-sums to detect abnormal behavior, and for storing passwords with salt.

A hash function must compute the hash in a short time, and the output must be the same every time for the same input. At the same time, a hash function must have specific security properties. Hash functions fundamental properties are described by Pre-image resistance, second Pre-image resistance, and collision resistance.

A hash collision occurs when different input strings produce the same hash. Hence we can attack the hash function with brute-force by trying for all possible combinations for Pre-image and Second Pre-image.

Hash functions properties:

- Pre-image resistance
  - Given a hash value H, it should be difficult to find any message m such that H = hash(m).
  - This means that it is not possible to find the hash input from the hash output.
- Second Pre-image resistance
  - Given an input messages m_1, it should be difficult to find a different input message m_2 such that hash(m_1) = hash(m_2).
  - This means that given an input and its hash, it should be hard to find a different input with the same hash.
- Strong Collision resistance
  - It should be difficult to find two different messages m_1 and m_2 such that hash(m_1) = hash(m_2).
  - This means that given two messages, there should be practically impossible to find two different inputs that produce the same input.

SHA family and RIPEMD Hashing algorithm

Secure Hashing algorithm 256 (SHA-256) is a member of the SHA-2 family. SHA-2 hashing algorithms published by NIST as a standard protocol and designed by the USA National Security Agency (NSA) in 2002. SHA-256 algorithm takes a message of 512-bits mixing with 256-bit hash code and a repeat round of 64 times, the output will be 256-bits hash. SHA-256 block size indicator is 64 byte, maximum allowed message length 33 bytes, the message digest size 32 bytes, word size 4 bytes, and the speed to generate approximately 140 MB per second [85]. SHA-256 is used in secure communications for web service and application as certificates for establishing and authenticated for a secure connection. It also used in several parts of the cryptocurrency Bitcoin network in mining and creating a Bitcoin address.

RIPEMD (Race Integrity Primitives Evaluations Message Digest) is a cryptographic hash function based upon the Merkle–Damgård construction. The Bitcoin standard uses the RIPEMD-160 version that reduces the chance of accidental collision. The compression function is made up of 80 stages and 5 blocks that run 16 times each. This pattern runs twice, with the results being combined at the bottom using modulo 32 addition.

SHA-3 (Secure Hash Algorithm 3) is the latest member of the Secure Hash Algorithm family, released by NIST on August 5, 2015. SHA-3 is internally different from the structure of SHA-1 and SHA-2. SHA-3 hash a sponge construction, which is based on a wide random permutation and allows inputting ("absorbing" in sponge terminology) any amount of data, and outputting ("squeezing") any amount of data, while acting as a pseudo-random function concerning all previous inputs.

3.6 Practical Use of Cryptosystems

As we mentioned earlier, cryptosystems are essential mechanisms to protect our data over the Internet and our systems from the intruders. Intruder attacks our system by using the Internet and wireless connections as a medium to gain access.

We will describe some security protocols that are used to protect the distributed network systems in this section, many of these protocols used by many systems for communication between the nodes for synchronization and authentication. Blockchain network also used some of these protocols below for communication and sending a secure transaction to the network nodes.

3.6.1 Public Key infrastructure

A public key infrastructure (PKI) is a framework for maintaining and distributing public-key certificates [76]. PKI supports authentication for entities such as people, devices, and services over the Internet. It enables controlled access to systems and resources, protects the system against unwanted eavesdroppers, it prevents impersonation of identity, and provides non-repudiation. The most widely used PKI is on the web, where trusted third parties called Certificate Authority (CA) certify authentication of public-key certificates. Digital certificate or public key certificates certifies the ownership of the public key. The X.509 standard specifies the format of certificates.

3.6.2 X.509 standard

X.509 is a standard defining the format of public-key certificates and is used in many Internet protocols. The ISO/ITU-T X.509 standard defines the X.509 public-key certificate or the X.509 attribute certificate. The X.509 public-key certificate is a digital certificate containing a public key for an entity and a name for that entity, together with some other information that is rendered un-forgeable by the digital signature of the certification authority that issued the certificate [76]. A public-key certificate, also known as a digital certificate is an electronic document used to prove the ownership of a public key. An X.509 certificate contains a public key, together with the owner's identity, and is either signed by a certificate authority or can be self-signed.

3.6.3 Transport Layer Security

Transport layer security (TLS) is a security protocol commonly used by Internet applications, such as Web browsers, Email services, Instant messages, etc. TLS is also known as HTTPS or HTTP + s. HTTP stands for hypertext transfer protocol, which is a protocol for communication between entities over the Internet or communication between a client and a server. HTTP is not a secure protocol for communication, so we have to implement the Secure Socket Layer (SSL) or TLS as a new security mechanism on top of TCP (Transmission Control Protocol).). TLS was initially being called SSL, and it changed the name after SSL version 3. The current TLS is version 1.3.

TLS provides privacy and data integrity communications security over networks and is a standard by the Internet Engineering Task Force (IETF) for using it as communication between client and server.

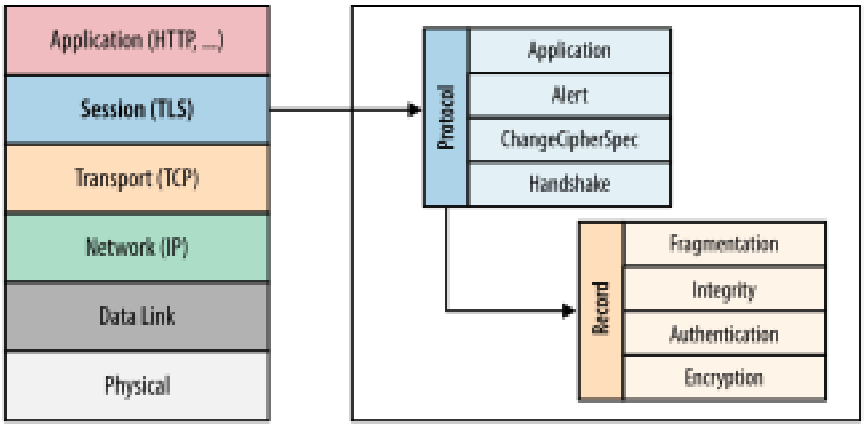


Figure 3.3: TLS [34]

The session establishment between client and server is called TLS handshake or the three-way handshake, where the server exchanges information with each other to decide on a secret shared key. TLS uses a cryptographic algorithm, Diffie Hellman, for the secret exchange of the key.

In the three-way-handshake protocol, the ChangeCipherSpec lets the client and server choose between different cipher suite algorithms that contain; Key exchange, signature, encryption cipher, and hashing algorithm. The TLS protocol also provides its message framing mechanism and signs each message with a message authentication code (MAC). Whenever a TLS record is sent, a MAC value is generated and appended for that message. The receiver can compute and verify the sent MAC value to ensure message integrity and authenticity [34].

3.6.4 IP Security and Media Access Control Security

Internet Protocol Security (IPsec) is a security protocol for protecting the IP packet by encrypting the data packet on the IP layer. IPsec provides data authentication, confidentiality, and integrity communication between networks. IPsec is a standard suite protocol issued by IEFT (Internet Engineering Task Force) for communication between two-point across the IP network[78].

IPsec is used in VPN (Virtual Private Network), VPN is used for connecting between two networks using IPsec protocols called Tunnel, for secure transport of data packets through the insecure Internet.

IPsec has two main protocols for providing traffic security services. These are (1) Authentication Header (AH) and (2) Encapsulating Security Payload (ESP). IPsec offers access control, enforced through the distribution of cryptographic keys and the management of traffic flows.

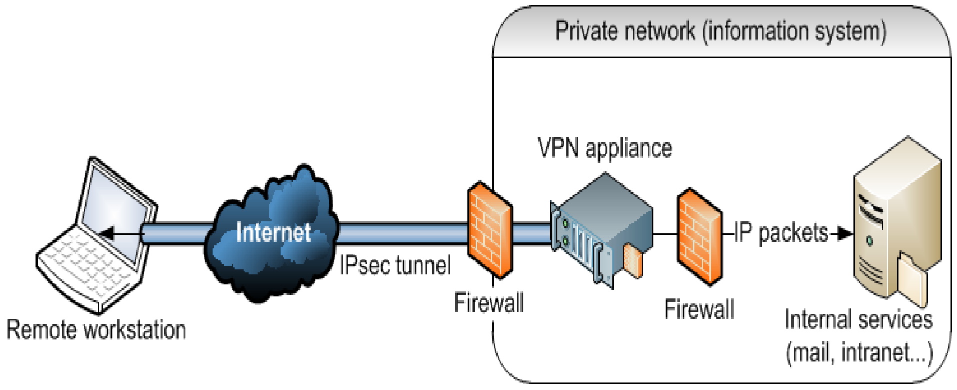


Figure 3.4: IPsec tunnel overview

IPsec has two tunnel modes that can be chosen; Transport mode and Tunnel mode. The Transport mode is typically used in end-to-end security, where only the payload of the IP packet is authenticated or encrypted. A hash secures the transport of IP Packet, so they cannot be modified. The Tunnel mode typically used in network-to-network, host-to-host, and VPN, where the entire IP packet is encrypted and authenticated into a new IP packet with a new IP header.

3.6.5 Domain Name System Security Extensions

Domain Name System Security Extension (DNSsec) builds on the Domain Name System (DNS), DNS was designed in 1980. DNS translates computer language to a recognizable human language domain names, such as IP-address iv4: 129.240.2.3, IP-addresses iv6: 2001:700:100:2:3 to Nissen.uio.no. DNS provides destination addresses on the Internet to a domain. DNS protocol translates a domain name into an IP-address without a strong authentication for the destination address.

DNSsec is implemented on top of DNS for the lack of DNS authentication [72]. DNSsec uses digital signatures based on public-key cryptography to solve the vulnerabilities of DNS. When we visit a domain such as uio.no, the uio.no homepage has a digital signature or fingerprint to prove that this is the original domain site. DNSsec will check for the authenticity of the DNS reply, and it checks if the hash signature is the same, the request will be rejected if the hash signature is not the same.

3.6.6 Border Gateway Protocol Security

The Border Gateway Protocol Security (BGPsec) is part of the Border Gateway Protocol (BGP) security mechanism. BGP is the connecting point of the Internet, just like a post office service, delivering mail. The BGP protocol looks at all of the available paths that data packets could travel through and chooses the best route to deliver the packet. Autonomous system (AS) exchange routing information between the AS system to find the best available paths without human interference using the BGP protocol.

A regular BGP router starts with a prefix, and it creates an AS Path that contains a local AS number. When a router nearby broadcast a BGP update messages, it adds the local AS number to the AS path, and the routers then determine whether this update exists in the AS path or not. It will reject an update message if it finds that it has the AS number already, so the routing loop does not occur.

The BGPsec [58] provides security for the BGP routing by replacing the AS path in BGP updates with a new BGPsec path, when two routes communication for sharing information, this is to make sure that the AS path is the legitimate, not a fake BGP Update message. The BGPsec generated a cryptographic signature for each of its neighbors to protects its integrity.

BGPsec relies on the Resource Public Key Infrastructure (RPKI) certificates that attest to the allocation of As number and IP address resources. The RPKI is a cryptographic method, also is known as Resource Certification, which is an infrastructure to support secure Internet routing. RPKI proves connections between specific IP address blocks or autonomous system number (ASN) with the others of Internet resource holders.

Part III

The third Part

Chapter 4

The impact of Quantum Computer on cryptography

In this chapter, we will describe the basic quantum computer, how the quantum computer can threaten the cryptography symmetric and asymmetric algorithm.

4.1 Quantum computer history

The discovery of fire was one of humankind's first and maybe the most significant invention. Fire keeps dangerous animals away, keeps us warm on a cold night, turns stone mineral to tools, and gave us time to evolve our creative brain.

In 1600 an English scientist William Gilbert wrote De Magnet, which describes static electricity by rubbing with certain materials. The conduct of electricity was further working on in the 17th and 18th centuries, where famous researcher such as Otto Von Guericke, Robert Boyle, Stephen Gray, C.F. du Fay, Benjamin Franklin, Alessandro Volta, George Ohm, Hans Christian Østed, Andre-Marie Ampere, James Clerk Maxwell, Thomas Edison, Humphrey Davy.

In around 1822-1840 an English mathematician Charles Babbage invented the first automatic digital computer called Difference Engine, the difference engine was capable of computing several sets of numbers. Ada Lovelace becomes the first programmer [64] that worker on programming the Difference Engine. The first functional modern computer Z1, an electro-mechanical binary programmable designed by Konrad Zuse, came in around 1936-1938. Around that time, Alan Turing purpose The Turing Maschine, it was a device printed following a series of logical instructions that created the fundamentals concept of a computer we have today.

A year earlier in 1935 Albert Einstein, Boris Podolsky, and Nathan Rosen published the paper " Quantum-Mechanical Description of Physical Reality Be Considered Complete? " this is known as the EPR paradox (Einstein, Podolsky, Rosen). Where Albert Einstein famous referring to quantum entanglement as "Spooky action at a distance". Albert Einstein did not like the concept of two particles that interact and then separate, because it seemed to violate the speed limit on the transmission of information from his paper on "Theory of relativity". Erwin Shroedinger experimented or created a theory by saying, " putting a cat with a bottle of poison and a radioactive source inside a box; when one looks inside the box, one could see the cat is either alive or dead." This was known as the Schrodinger's cat paradox. It describes the quantum state known as superposition.

In early 1980 Richard Feynman urges the world to build a quantum computer in a conference hosted by MIT (Massachusetts Institute of Technology) and IBM (International Business Machines). Where he states the possibility of using quantum effects for computation in his lecture " There's Plenty of Room at the Bottom: An Invitation to Enter a New Field of Physics ". Two years later, 1984, after the conference, IBM scientist Charles Bennett and Gilles Brassard publish the world first quantum cryptography protocol BB84. David Deutsch describes the first universal quantum computer able to simulate any other quantum computer with at most a polynomial slowdown in 1985.

In 1994 Peter Shor's algorithm introduced a way that could break the hard mathematics problem or computational problem behind the public-key cryptography, solving the factorization problem and the discrete logarithm problem. In 1996 Lov Grover introduce the quantum database search algorithm that has quadratic-speedup brute-force attack on cryptography keys.

In 2009, researchers from Yale University created the 2-qubit superconductor chip made of a billion aluminum atoms that acted like a single atom that could occupy two states., this was the world first solid-state quantum processor, in 2011 team of scientists from Australia and Japan successfully transferring a set of quantum data with full transmission integrity, without affecting the qubits superpositions. In the same year, a Canadian company D-Wave Systems announced the first commercial quantum computer with a 128-qubit processor.

In 2013 Google was launching the Quantum Artificial Intelligence Lab via partnerships with NASA Ames Research center with a D-Wave quantum computer.

In 2016 IBM Research announced making five qubits quantum computing available for everyone by putting out on the cloud service IBM Quantum Experience for everyone who wants to test, run and experiment with how a quantum computer works. In the same year, scientists at the University of Maryland successfully built the first re-programmable quantum computer]. China launches a quantum communication satellite using Quantum Key Distribution(QKD). QKD's goal is trying to make communication safer by using quantum entanglement principles to confuse eavesdropper

4.2 Quantum mechanics

Quantum mechanism or Quantum theory is a theory that explains the nature and behavior of the energy element on a microscopic level. In 1900 Max Planck, a theoretical physics, presented the quantum theory to the German Physical Society. Max Planck's hypothesis that energy is radiated and absorbed in discrete energy packet, precisely matched the observed patterns of black-body radiation. The Black body radiations are a matter that absorbs and emits all electromagnetic radiation that is coming in contact.

Planck constant:

$$E = h * v, \text{"where h is Planck constant."}$$

Later, Albert Einstein introduces the photoelectric effects, where electron or light reflects when hitting a material. Max Planck and Albert Einstein form a basic understanding of energy behavior for modern physics.

4.2.1 Quantum phenomenal

In a classical computer, one bit of binary data is called bits. Bits can be "1" or "0". In a quantum computer, there are quantum bits or qubits. Qubits are bits that can simultaneously be one and zero at the same time; Qubits in this state, we called a superposition state. The superposition is like flipping a coin in the air; when it is in the air, it gives a probability of both head and tail. When it lands on a material, it has 50% to become head and 50% tails.

A quantum computer uses superconductor electronic circuits to achieve the superposition. A superconductor is a specific material that has a characteristic to conduct electricity or transport electrons from one atom to another with no resistance. The superconductor becomes superconductive when it reached a critically low temperature.

A superposition needs to be in a cold environment that is near a freezing point of absolute zero. Absolute zero is around -273 Celsius to -273,5 Celsius. When a quantum computer is in a superposition state, the state is sensitive to noise and frequency from the outside world, so the quantum computer needs to be built inside an isolated environment to keep all interference noise and frequency out. An entanglement between electrons is called quantum entanglement. The quantum entanglement is an invisible thread that keeps the electron connected with each other. Knowing one state of an electron, we will have an idea of what states the other one will be.

An example of a pair of gloves that describes the concept of quantum entanglement [89];

If we found a right-handed glove, then we automatically assume that the left-handed gloves are missing. The left-handed gloves could be found from a long-distance faraway from the right-handed gloves. If we did not found the left-handed *gloves, both gloves could be left-handed or right-handed, until the moment we observe and found one of them. If we never found the right-handed gloves, we never knew that the left-handed were missing .*

4.2.2 Basics quantum mathematics and notation

Quantum physics describes particle movement as spin; a spin is an angular force generated by a current flow of energy in the probability wave field of the particle. A spin can be measured by either spin-up or spin-down align with the direction of measurement. We can use the spin measurements to describes the behavior of the superposition with notation.

Superposition linear combination of both: α and β is the amplitude of measuring the value.
$|0\rangle$ is the spin-up: $\uparrow$
$|1\rangle$ is the spin-down: $\downarrow$
The first measurement can be either spin-up or spin-down, lets said spin up is: $\alpha_0|0\rangle$.
The second can also be either spin-up or spin-down, lets said spin down is: $\alpha_1|1\rangle$.
Then the quantum state $|\Psi\rangle$ can be written [36, 79, 92]:

$$|\Psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

We can write this as basic matrix notation:

$$spinUp : |0\rangle = \begin{vmatrix}1\\0\end{vmatrix} spinDown : |1\rangle = \begin{vmatrix}0\\1\end{vmatrix}$$

2-qubits with two spin particles will look like this: $|\Psi_0\rangle$ first qubit, $|\Psi_1\rangle$ second qubit.

$$|\Psi_0\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$
$$|\Psi_1\rangle = \beta_0|0\rangle + \beta_1|1\rangle$$

The product of 2 qubits $|\Psi_0\rangle\,|\Psi_1\rangle$:

$$|\Psi_0\rangle\,|\Psi_1\rangle = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle$$

As we could see, 2 qubits give us 00, 01, 10, 11. it gives us 4 computations power, and we could say the equation is 2^n, where n is the number of qubits. 3 qubits will give us 2^3 that will be 8 computational power. For 32 qubit we will have $2^{32} = 4,294,967,296$ computational power. For every qubit, it will give an exponential growth of computational power. The only problem with a quantum computer is that it is hard to understand and observe what is going on inside the qubits states. When it is in the superposition states, we can only measure the result of it.

Quantum gates

Quantum computer does not use basic operators NOT, NAND, XOR, AND, OR to build complex operations like classical computers. Quantum computes us basic operators called quantum gates such as Hadamard gate, Pauli-X gate, Pauli-Z gate, Phase shift gates [79, 87, 92]. Since it is hard to understand how inside superposition operates. Quantum computers use different gate algorithms to manipulate the result of the qubits. Quantum computers use different gate algorithms to manipulate the result of the qubits. We manipulated theta θ and Phi Φ of the superposition to move along the surface of the unit sphere [36].

Different between a classical computer and a quantum computer is that the operators in a quantum computer can be reversible. Quantum gates have to be designed so that there is another quantum gate that can reverse quantum state back to its original.

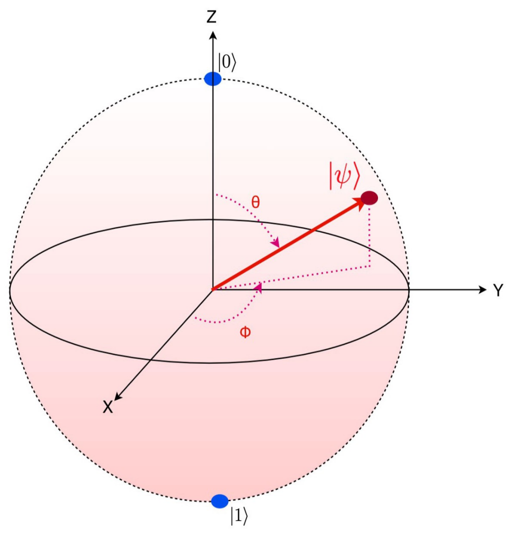


Figure 4.1: The Block sphere [36, 79]

The Block sphere

The Block sphere is named after physicist Felix Bloch; it a representation of the quantum mechanical. The value theta θ and phi Φ cover the whole sphere, $\theta \in [o, \pi)$ and $\Phi \in [0, 2\pi)$. We can say that angle θ corresponding to the latitude and angel Φ corresponding to longitude [52]. π is the ratio of a circle circumference (c) to its diameter (d) $\pi = \frac{c}{d}$. The represent rotation of π is:

$$\frac{(x+z)}{\sqrt{2}}.$$

An sphere have x,y and z, where Radius r are define by $r^2 = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2$.
We can find the points on the sphere with this formula [88]:

$$x = x_0 + r\sin\theta\cos\Phi$$
$$y = y_0 + r\sin\theta\sin\Phi$$
$$z = z_o + r\cos\theta$$

Sphere volume and areal:

$$V = \tfrac{3}{4}\pi r^3 \text{ and } A = 4\pi r^2$$

We can use the basics of mathematics on the sphere as a basic to manipulated the qubits state in the sphere to collapse the direction we want to measure it.

We can write superposition as $|\psi\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad |\alpha|^2 + |\beta|^2 = 1$$

Hadamard Gate: [36, 79, 87, 92]

The Hadamard gate is the first part of the algorithm to make a reversible gate back to its original. Basic of quantum states spinUp and spinDown can be written like a rotation of Pi π:

$$|0\rangle \to \tfrac{|0\rangle+|1\rangle}{\sqrt{2}} \text{ and } |1\rangle \to \tfrac{|0\rangle-|1\rangle}{\sqrt{2}}$$

The Hadamard H acts on a single qubit gate can be written:

$$H = \tfrac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \text{ or } H = \tfrac{|0\rangle+|1\rangle}{\sqrt{2}} + |1\rangle\,\tfrac{|0\rangle-|1\rangle}{\sqrt{2}}$$

Hadamard gate implemented on block sphere:

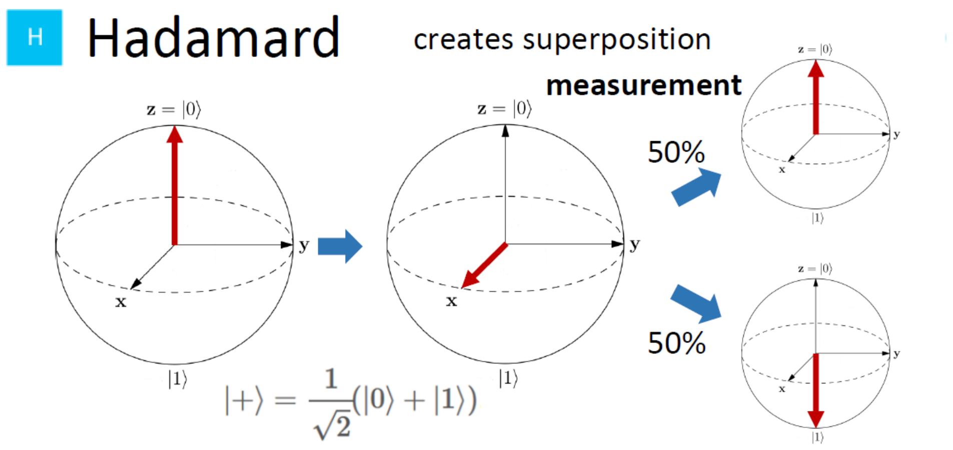


Figure 4.2: Hadamard gate superposition.

Pauli Gate [36, 79, 87, 92]

Pauli gate has 3 functions gates; X-gate, Y-gate, and Z-gate. This gate coordinates the rotation of the superposition along x,y, or z-axis. The X-gate rotated the vector along the x-axis. Describes as the classical operation"NOT gate" because it changes the amplitudes of the superposition from $|0\rangle$ to $|1\rangle$ or $|1\rangle$ to $|0\rangle$.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Y-gate rotated around the y-axis: It switches the amplitudes and multiple then with $\pm i$.

$$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

Z-gate rotated a single qubit by π around the Z-axis:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Phase Shift rotates the vector around the z-axis that defines by names. T-gate, S-gate, and phase shift gates.

$$\text{S-gate} \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}. \text{ T-gate} \begin{bmatrix} 1 & 0 \\ 0 & \exp^{i\pi/4} \end{bmatrix}. \text{ Phase shift gate } (R_\theta\ ) \begin{bmatrix} 1 & 0 \\ 0 & \exp^{i\theta} \end{bmatrix}$$

4.2.3 Lov Grover algorithm

Grover's algorithm published in 1996 by L.K. Grover. The second most famous quantum algorithm after Shor's algorithm, Grove's algorithm solves the problem of unstructured search on a quantum computer in $O(\sqrt{N})$ operations or set of $N = 2^n$, while the fastest algorithm on unstructured search in classical computer O (N) time. This means with Grover's algorithm, we can search a database queries $2^{n/2}$ or $\sqrt{n}$ faster than a classical computer. Grover's algorithm can only search a database that's is in a quantum superposition state.

Example of how Grover's algorithm in theory [36, 79, 87, 92]:

1. Initial state : $|0\rangle^{\otimes n} = |0\rangle$.
    - Where n is the number of qubits necessary to represents the search space of size $2^n = N$.
2. Apply the Hadamard transform to all qubits:

$$H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle = |\psi\rangle.$$

- Hadamard transform equalizes the superposition states in the system.
- Requires operation $\Theta(log\, 2^n)$.

3. Apply the Grover iteration: $R \approx \frac{\pi}{4}\sqrt{2^n}$ times.

$$[(2|\psi\rangle\langle\psi| - I_n)]\,|\psi\rangle \;\approx |x_0\rangle$$

4. Measure $|x_0\rangle$ the register output.

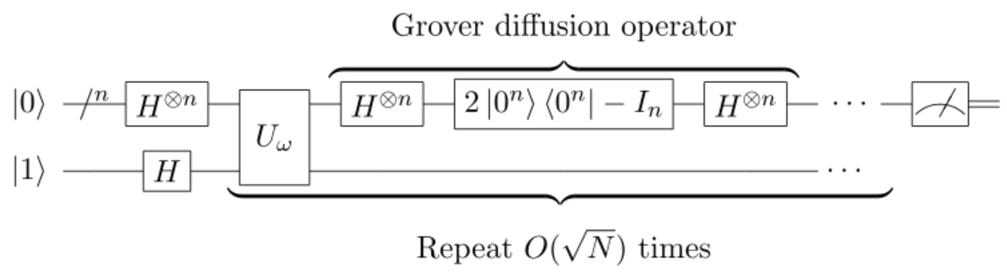

Grover diffusion operator
$|0\rangle$
n
$H^{\otimes n}$
U_ω
$H^{\otimes n}$
$2|0^n\rangle\langle 0^n| - I_n$
$H^{\otimes n}$
$|1\rangle$
H
Repeat $O(\sqrt{N})$ times

Figure 4.3: Grover's algorithm [86]

4.2.4 Peter Shor's algorithm

In 1994 Peter Shor found a way to solve the factorization of large integers and discrete logarithms in polynomial time. Peter Shor algorithm was inspired by Daniel Simon algorithm, the first algorithm to have exponential speedup over the classical algorithm [79]. Peter Shor algorithm prove that full-scale quantum computer would have potential to threat the cryptosystem. Shor's algorithm shows that quantum computers can be used to factor a number n in polynomial time breaking [49] RSA and elliptic curves.

4.3 Quantum resistant cryptography

Quantum resistant cryptographic is believed to be resistant against quantum attacks. Lattice-based cryptography, code-based cryptography, multivariate-quadratic equations cryptography, and hashed-based cryptography is quantum resistant cryptography. NIST has a call for proposals post-quantum cryptography started around 2017 [68] announced in 2.April 2016 and the end date for post-quantum cryptography proposals has not been set.

We will focus on hash-based quantum-resistant cryptography in this thesis and try to propose a signature schemes replacing digital signature ECDSA for cryptocurrency Bitcoin. Below we introduce a quick summary of some blockchain that has implemented a quantum-resistant signature and papers that introduce how to make blockchain and Bitcoin more quantum robust.

- **Quantum Resistant Ledger (QRL):** Is a proof-of-stake blockchain that has implemented a quantum-resistant signatures scheme on its blockchain systems. QRL chooses the hash-bashed one-time signature Winternitz OTS+ (W-OTS+) scheme for its digital signatures and eXtended Merkle Signature Scheme (XMSS) with QRL hypertree. QRL using SHA-256 as the cryptographic hash algorithm. QRL blockchain can be download and tested on [81].

- **Blockchained Post-Quantum Signatures (BPQS):** Wrote by Corda (R3) team. BPQS is a single-chain variant of XMSS family protocols, it focuses on short and fast one-time signatures with the extra option to re-design if needed. The BPQS scheme use of L-Trees and generation of bitmasks to define the security assumptions and proofs. BPQS design can combine with other one-time signature to make it more secure. BPQS is a prototype that can be seen here [80]. We thought of an idea after reading; If the BPQS design should have used OTS signature on signing the block, that has been done writing to the chain. Than we will have another layers to protects the previous block from quantum attack.

- **Committing to quantum resistance: a slow defence of Bitcoin against a fast computing attack**. This paper [40] proposed a soft-fork commit-delay-reveal scheme to allow for the secure transition to a quantum-resistant address scheme in Bitcoin and propose a time frame period of six months for client and miners can use their address.

- **Quantum-secure Blockchain:** Wants to use quantum key distribution (QKD). QKD can generate a secret key between two parties connected by the quantum channel for transmitting in a quantum state, where it is distributed using a fiber network that has a distance of 421 km [57].

  This paper [28] proposes to create a new blockchain with QKD as a distribution of keys and a broadcast protocol on how to deliver the key.

- **Quantum attacks on Bitcoin, and how to protect against them:** This paper [24] investigated a better alternative to make Bitcoin Proof-of-Work algorithm mining protocol more robust against attacks and less electricity consumed by suggestions change to Equilhash, Momentum and Cuckoo Cycle. Equilhash, based on the generalized birthday problem. Momentum, based on finding collisions in a hash function, and Cuckoo Cycle, based on finding constant sized subgraphs in a random graph.

- **IOTA:** Is a distributed ledger technology (DLT) that allows devices in an IOTA network to transact a micropayments and to send each other immutable data. IOTA used the W-OTS (Winternitz one time signature) to generate a digital signature. IOTA addresses can only use one time, if uses more than one time security of the address decrease significant [41].

4.3.1 Hashed-Based Digital signatures

Hash-based signatures quantum-resistant rely upon cryptographic hash function one-way security, which combined with Merkle tree structure. Hash-based signatures require a public key (pk) for verification and a private key (sk) for signing a message.

Hash-based digital signatures cryptography using hash functions as their main encryption tools. The benefits of using hash-based signature schemes are that, it is easier to replace a hash function with another more secure hash function without do big changes in the system, if a flaw were to discover.

Lamport One-time signature (L-OTS)

In 1979 Leslie Lamport described Lamport signatures scheme or Lamport one-time signature schemes a method for constructing a digital signature, where the key can only be used to sign a single message. Using the key more than one time makes the key less secure, and some might re-create the signature from the shared message.

Let us demonstrated Lamport one-time signature using a hash function that takes in 256-bit inputs and gives out 256-bit output. Creating a private key (sk) and a public key (pk).

First, we produce 256 pairs or 512 random bit-strings, each number has 256-bits in size. These will be our sk, total size of 256 pairs: 256*256* 2 = 131 072 bits or 128 kb. Furthermore sk will be written as a list of sk_0 and sk_1.

$$sk_0 = sk_1^0, sk_2^0, sk_3^0, sk_{256}^0$$
$$sk_1 = sk_1^1, sk_2^0, sk_3^0, sk_{256}^0$$

We then hash each random number from sk to produce the list pk_0 and pk_1. We now can broadcast list pk_0 and pk_1 to the world:

$$pk_0 = H(sk_1^0), H(sk_2^0), H(sk_3^0), H(sk_{256}^0)$$
$$pk_1 = H(sk_1^1), H(sk_2^1), H(sk_3^1), H(sk_{256}^1)$$

Signing a message with Lamport OTS (L-OTS): First, we hash the message (M) with 256-bit using sk. The hashed message will be $M_1, M_2, M_3, .., M_{256}$. Based on the value of the bits in the hash message, we pick a number from the sk list until we reach 256 numbers, these numbers will be our signatures. The way we pick numbers from sk is based on a bitwise inspection of sk list sk_0 and sk_1. If the message M_1 contains bit 0, we pick a number from sk_0 in the order the number stands, M_1 pick position sk_0^1. If M_2 contains bit 1, we pick a number from sk_1 in the order the number stands, M_2 pick position sk_2^1.

We can write a notation: $M_i = 0$, we pick sk_i^0. $M_i = 1$, we pick M_i^1. $i = (1, 2, 3, .., 256)$, and it represents the order where the number stands. We will do this until we reach 256 number, these 256 will be our signatures (s). This produces a signature size of total 256*256 bits = 64 KB. The messages will be sent with the signatures.

Verifying the signature of the message: First, we check the messages M_i bit, and we hash signatures (s_i). List pk_0 and pk_1 are known, so we compare the hash (s_i) with the two pk list pk_0 and pk_1, if the $M_i = 1$, we pick list $pk1$ and the order it stands. If $M_i = 0$, we pick list pk_0. We will see that hash (s_i) is the same as the public key(sk) list.

Winternitz OTS

Robert Winternitz introduced Wintersnitz one-time signature scheme (W-OTS) and, in 2011, further optimization by Buchmann, Dahmen, Ereth, Hülsing, and Rückert [47]. W-OTS uses small key and signatures sizes are believed to be quantum-resistant. The size of a Winternitz signature is roughly mn/w bits and signing roughly requires $2^w m/w$ hash operations, where m is the bit length of the hash value to be signed, n is the output length of the hash function used in the scheme, and w is the Winternitz parameter determining the trade-off between signature size and signature generation time [47].

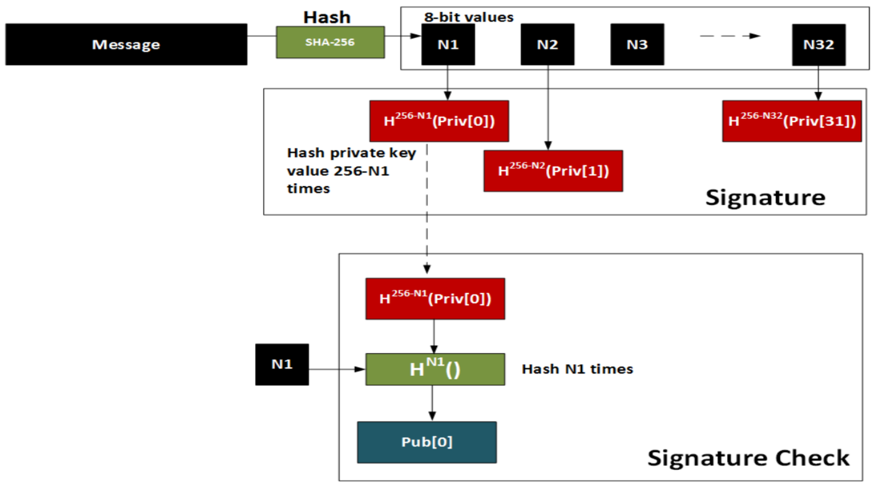


Figure 4.4: W-OTS sign and verification process [16]

Winternitz OTS created a pair of keys: a private key (sk) and a public key (pk). To created sk we generate a random list of 256 seeds: $sk_0 =$

$(sk_1^0, sk_2^0, sk_3^0, sk_{256}^0)$. We have to define the Winternitz parameter w and hashed the sk with 2^w. Lets us said $w = 8$, we will then have 32 value of sk. We hashed each number on the list sk_0 for 255 times, the hash number 256 $(H(sk_{256}^0))$ will be our pk.

Signing a message with W-OTS given that is hashed with SHA-256. Since $w = 8$, we will have 32 signature number $s_i = 32$ and $s_i = (N1, N2, N3, .., N32)$ and each number has a size of 8-bit. where N_i is calculated by H^{256-N1} and the position of where the number in list $sk[0]$. So if $N1 = 15$, we first have $H^{256-11} = H^{245}$, we take the hash of sk_{245} and from that list, we take the number in the position that N_i stands. If $N_i = 1$ takes position number 0, if $N_i = 20$ is takes the position number 19.

To prove the signature, we take the message and hash it with SHA-256, and then we take the signature s_i and hashed the amount times as the signatures number, we will arrive at the public key.

Winternitz one time signature scheme:

Algorithm 4: Winternitz OTS Key Pair Generation [48]

```
System Parameters: hash function H: ${0,1}^* \to {0,1}^s$, parameters
  $w \in \mathbb{N}$ and $t = [s/w] + [(log_2[s/w] + 1 + w)/w]$
Output: Signature key $X$, verification key $Y$
1: choose $x_1, ..., x_t \in_R \{0,1\}^s$ uniformly at random.
2: set $X = (x_1, ..., x_t)$.
3: compute $y_i = H^{2^w-1}(x_i)$ for $i = 1, ..., t$.
4: compute $Y = H(y_1||...||y_t)$, where || denotes concatenation.
5: return $(X, Y)$.
```

Algorithm 5: Winternitz OTS Signature Generation [48]

```
System Parameters: hash function H: ${0,1}^* \to {0,1}^s$, parameters
  $w \in \mathbb{N}$ and $t = [s/w] + [(log_2[s/w] + 1 + w)/w]$
Input: document $d$, Signature key $X$
Output: one-time signature $\sigma$ of $d$
1: compute the $s$ bit hash value $H(d)$ of document $d$.
2: split the binary representation of $H(d)$ into $[s/w]$ blocks
  $b_1, ..., b_{[s/w]}$ of length $w$, padding $H(d)$ with zeroes from the left if
  required.
3: treat $b_i$ as the integer encoded by the respective block and
  compute the checksum
      $$C = \sum_{i=1}^{[s/w]} 2^w - b_i \quad (4.1)$$
4: split the binary representation of C into $[(log_2[s/w] + 1 + w)/w]$
  blocks $b_{[s/w]+1}, ..., b_t$ of length w, padding c with zeroes from the
  left if required.
5: treat $b_i$ as the integer encoded by the respective block and
  compute $\sigma_i = H^{b_i}(x_i), i = 1, ..., t$ where $H^0(x)$ := x.
6: return $\sigma = (\sigma, ..., \sigma_t)$.
```

Algorithm 6: Winternitz OTS Signature verification [48]

```
System Parameters: hash function H: $\{0,1\}^* \rightarrow \{0,1\}^s$, parameters
  $w \in \mathbb{N}$ and $t = [s/w] + [(log_2[s/w] + 1 + w)/w]$
Input: document $d$, signature $\sigma = (\sigma, ..., \sigma_t)$, verification key Y
Output: TRUE if the signature is valid, FALSE otherwise
1: compute $b_1, ..., b_t$ as in Winternitz signature generation
  Algorithm 4.
2: compute $\phi_i = H^{2^w-1-b_1}(\sigma_i)$ for $i = 1, ..., t$.
3: compute $\Phi = H(\phi_1||...||\phi_t)$.
4: if $\Phi = Y$ then return TRUE else return FALSE
```

In 2013 Andreas Hulsing introduced W-OTS+ [2] a further improvement of W-OTS with a checksum using bitmask XOR to chain functions of W-OTS.

Chain function $c_k^i(x, r)$: x on input of $x \in (0, 1)^n$, iteration counter $i \in N$, k is the key, and randomization elements $r = (r, ..., r_j) \in 0, 1^{n*j}$ with $j \geq i$ is define follows:

- W-OTS: $c^i(x) = h_k(c^{i-1}(x))$
- W-OTS+: $c^i(x) = h_k(c^{i-1}(x) \oplus r_i)$

4.3.2 SPHINCS+

SPHINCS+ arrives from SPHINCS scheme [23] develop by team from Department of Computer Science, University of Illinois at Chicago.

SPHINCS+ Schemes uses XMSS (Extend Merkle Signatures Schemes) called $XMSS^{MT}$ [3], a hypertree that made of a large number of binary hash trees, for signing a large but fixed number of messages. The top of a hypertree used to sign the root nodes of the tree or leaf from the layer below. Trees or leaves on the lowest layer are used to sign the messages. $XMSS^{MT}$ trees have equal height, where the total height h that has layers d of XMSS trees. We can say XMSS trees of height: h/d, and layer $d - 1$ contains one XMSS tree, layer $d - 2$ contains $2^{h/d}$ XMSS trees, and the final layer 0 (zero) contains $2^{h-h/d}$ XMSS trees. XMSS tree has a 2-leaf node child associated with a hash string that concatenates with each other, and this will make the tree to grow in height and width. In each of the leaf child nodes, there will be public keys of the Winternitz one time signatures scheme (WOTS+). The public keys are used as sub-tree leaves, where it is corresponding with the private keys on the sign-roots of lower-layer sub-trees.

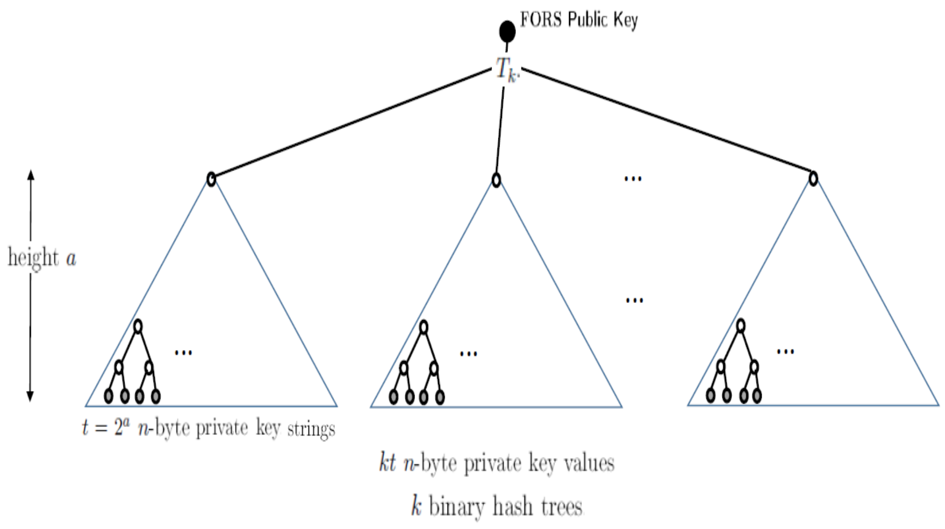

FORS Public Key
T_k
height a
$t = 2^a$ n-byte private key strings
kt n-byte private key values
k binary hash trees

Figure 4.5: FORS Tree and Public key [45]

On the lowest layer, we have a few-time signature (FTS) scheme FORS (Forest of Random Subsets). FORS is an improvement of HORST [23] from SPHINCS. FORS is defined in terms of parameters n, k, and $t = 2^a$. FORS used to sign strings of length ka bits. The $SPHINCS^+$ parameters are described below [45].

- n: The security parameter; it is the length of a private key, public key, or signature element in bytes.
- k: The number of private key sets, trees, and indices computed from the input string. The number of trees in FORS.
- t: The number of elements per private key set, number of leaves per hash tree and upper bound on the index values. The parameter t MUST be a power of 2.
- If $t = 2^a$, then the trees have height a, and the input string is split into bit strings of length a. t is the number of leaves of a FORS tree.
- a is $log\ t$.
- m: The message digest length in bytes. It is computed as $m = b(k\ log\ t + 7)/8 + b(h - h/d + 7)/8 + b(h/d + 7)/8$.
- len: The number of n-byte string elements in a $WOTS+$ private key, public key, and signature.
- w : The Winternitz parameter. An element of the set {4, 16, 256}.
- h : The height of the hypertree.
- d : The number of layers in the hypertree. signature

FORS generated pseudorandom private key values by using a private seed SK.seed from $SPHINCS+$ private key. FORS private key values are only temporarily generated for computing the public key or a signature. The FORS private key consists of kt random n-byte strings grouped into k sets, each containing t n-byte strings. The FORS public key is a single n-byte hash value. It is computed as the tweakable hash of the root nodes of k binary hash trees. Each of these binary hash trees has height a and is used to authenticate the t private key values of one of the k sets. A tweakable hash function takes a public seed PK.seed and context information in the form of an address ADRS in addition to the message input.

In $SPHINC+$, a private key scheme has two randomly generated seed. The first generates FORS private key value, and the $wots^{+}$ private key. The second generated a randomization value (OptRand) for the message hashing. There are two public key ones for the root (PK.root) in the top layer for signing messages and the second public key (PK.seed) for FORS to sign and create $wots^{+}$.

$SPHINCS^{+}$ signature consist of: Randomness R (n bytes), FORS signature SIG_{FORS} ($k(a+1) \cdot n$ bytes) and hyper-tree (HT) signature SIG_{HT} ((h + $d$len) n bytes).

$SPHINCS+$ signature generation follows 4 steps:

1. We generated Random value R, used to compute a, m byte message digest.
2. The value messages digest (md), idx_tree (index tree), and idx_leaf (index leaf) are computed by extracting the specified number of bits.
3. The partial md will be signed with idx_leaf FORS key pair and idx_tree from the XMSS lowest layer.
4. The public key of the FORS key pair is then signed using HT.

$SPHINC+$ verification: Generated FORS public key, and Compare the public key with a hyper tree (HT).

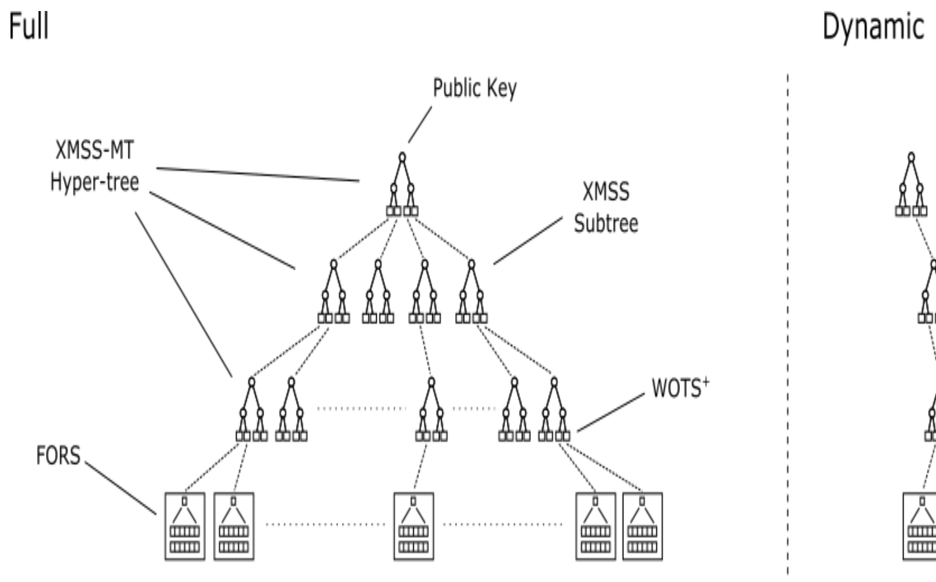


Figure 4.6: $SPHINC^+$ Full architecture [27]

4.4 Adding Post-Quantum Signatures to Bitcoin

4.4.1 Cryptographic Keys and Addresses on Bitcoin

Everyone who makes transactions with Bitcoin has a Bitcoin address. A Bitcoin address derives from an ECDSA public key going through Base58check encoding. An ECDSA Public key derives from an ECDSA private key through elliptic curve cryptography, namely ECDSA (Elliptic Curve Digital Signature Algorithm). An ECDSA private key is a random number created by a random point in the elliptic curve Sec256k1.

Algorithm 7: Private key to Public key

System Parameters: Private key k, Generator Point G, Public key K, ECC-Point(x,y).

1. We generate a 32-byte random number from 1 to 2256. This number should be generated from a secure random number generator. We can use a TRNG (True Random Number generator) or PRNG (Pseudo Random Number Generator) or a PRF (Pseudo-Random Family) to generate the private key and add to ECC. The Bitcoin uses ECC started point as the private key, and any secure random number can put in ECC.

2. We take the 32-byte random number as our private key k. The generation point G is a random start point in the elliptic curve secp256k1.

3. The point coordinates present public key K in ECC point (x,y), where $K = G * k$.

Algorithm 8: Public key to Bitcoin Address

System Parameters: Private key k, Generator Point G, Public key K

1. Take the public key K and hashes it with SHA-256: SHA-256(K).
2. We hash the above SHA-256 again with RIPEMD-160: RIPEMD-160(SHA-256(K)).
3. Now we have public key Hashed for verification and authentication. The Hash SHA-256 gives us 32-bytes, hashed with RIPEMD-160 again gives us 20-Bytes of data. That means the RIPEMD-160 hash does not contain the full public key, and it is impossible to find the full public key from RIPEMD-160 hashed.
4. Take the public key hashed through the Base58Check Encoding mechanism explained below.
5. Add the Version byte in front of RIPEMD-160 or SHA-256, (0x00 for Main Network,04x00 for the original format, compressed public key 02x00 or 03x00). A compressed public key is a public key that has only one ECC-point x-value. The ECC-point below y-axis is odd and will give 03, the above y-axis is even and given 02. The Version byte is a 4 bytes size and defines the type of data format
6. Take the public key, which is a double hash, and compute with Base58check encoding to make it human-readable, and based58check encoding is explained in the section below.
7. The output from Base58check is the Bitcoin address.

4.4.2 Base58 and Base58Check Encoding

Satoshi Nakamoto introduced Base58 as a modified version of Base64. Base64 is a set of 64 characters chosen to represent 64 digital binary values as human-readable text. Base64 strings have alphabetic characters (A-Z, a-z), digits (0-9) and symbols (+ and /), where Base58 is the same but without zero "0", lower case "L", capital "I", Capital "O" and non-alphanumeric characters plus (+) and slash (/). Base58 was introduced to remove the confusion of the number, symbol, and character when one has to copy the address, code to another location, printing or typing in Bitcoin address.

Base58Check encoding is an error detection method format, it is used to encode the hashed payload ECDSA public key, hashed reeddemScript, and ECDSA private key for wallet import format. The encoding started with Version bytes, a prefix data that serves to identify the type of data is encoded. The Version byte is concatenated with the payload and double hash. The checksum takes the first 4 bytes of 32-bits from the last double SHA-256-hashed. These 4 Bytes are added to the payload (public key hash), which then becomes a 25-byte address. The checksum helps to ensure that the received data is correct and not corrupted. Base58 encodes a binary string to a human-readable Bitcoin address [84].

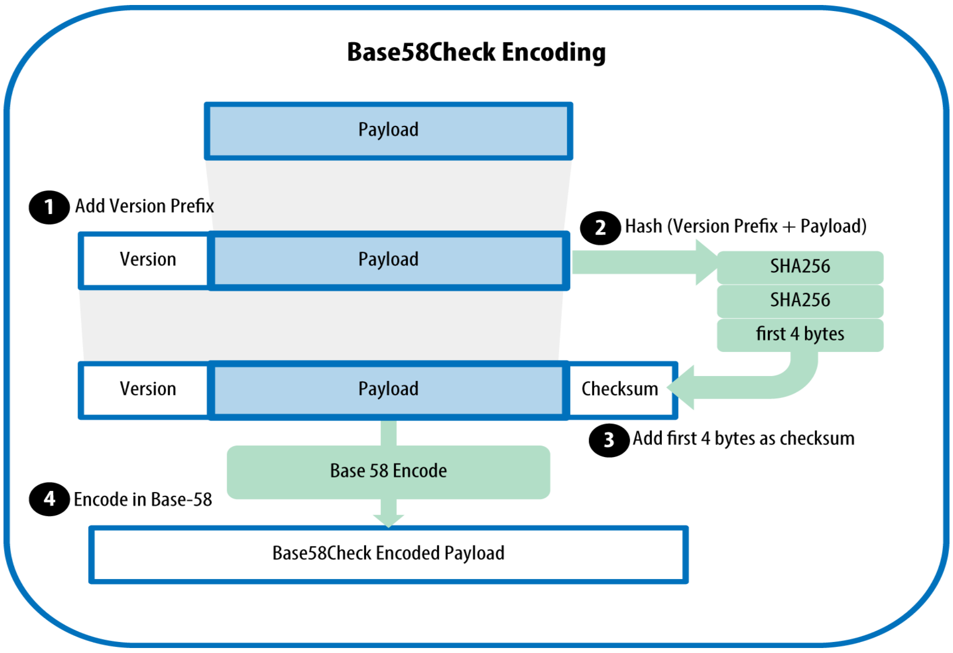


Figure 4.7: Base58Check Encoding [4]

4.4.3 Bitcoin Transaction and Validation

We have written little about how a transaction payment works in section 2.3.1 Bitcoin page 18-19. We will now go in more detail on how the coin or money in Bitcoin can be verified to originate from the owner of the public key or Bitcoin address without showing the private key.

A transaction is a payment mechanism that lets users sell and buy using Satoshi as currency, and Satoshi is a Bitcoins currency when we talk in small value. For example; 1 US Dollar is equal to 100 US cents, 1 Bitcoin is equal to 100,000,000 Satoshi. Each transaction has at least one output and one input. Each transaction output waits as Unspent Transaction Output (UTXO). UTXO is our balanced funds in the Bitcoin currency recorded on the blockchain and recognized as a currency unit by the network. If we have 1 Bitcoin or 100 million Satoshi coin in our balance, that means we have 1 Bitcoin waiting in one or more waiting in our unspent UTXO for spending.

For example:

Alice has 100 BTC and sends 50 BTC to Bob. Before Bob can receive 50 BTC, the transaction has to be validated by the Bitcoin network. Alice created a standard P2PKH transaction output containing instructions that allow anyone to spend that output if they can prove they control the private key corresponding to Bob's hashed public key. These instructions are called ScriptPubkey or pubkey script. Alice broadcasts the transactions to the Bitcoin network. The network checks if Alice has enough funds for transactions and is the rightful owners of that address, by checking all Alice's previous old transactions, and UTXO output that contains her public key hashed for validation. The Bitcoin network then created two outputs: one output with 50 BTC to sends to Bob's address and one to Alice's address for was is left of her funds. Bob's wallet now displays 50 BTC spendable balance. Bob later want to send 25 BTC of his 50 BTC to Steve. He must create an input which references to a transaction identifier (txid), pointed to the output index transaction Alice has created. Bob then must create a signature script or scripting that satisfies the conditions Alice placed in the previous output pubkey script or ScriptPubkey. ScriptPubkey and scriptSigs combine the pubkeys and signature inside a P2PKH script for authorization and verification. The P2PKH script is a locking script using a FORTH-like language. The FORTH-like script is stack-based and takes one operation time.

Bob's scriptSigs contains two pieces of data: One is ECDSA public key, so the pubkey script can check that is hashes the same values as the pubkey hash Alice has provided. Second is the digital signature, which proves Bob's ownership of that public key. When the check for validation of the script becomes true, Bob broadcasts the transactions that he wants to send 25 BTC to Steve's address. Steve wants to send 10 BTC to Kate. He has to go through the same process that Bob's did, when Alice sends Bob the 50 BTC.

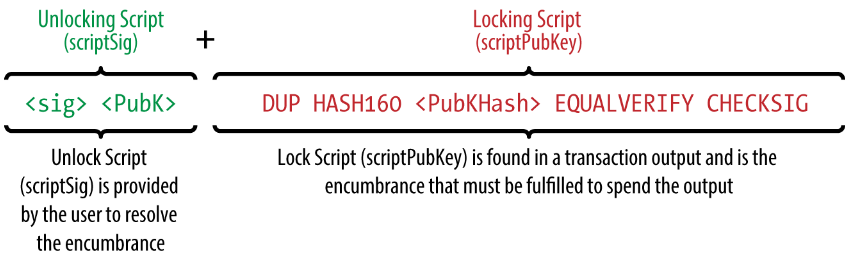

Unlocking Script
(scriptSig)
+
Locking Script
(scriptPubKey)
<sig> <PubK>
DUP HASH160 <PubKHash> EQUALVERIFY CHECKSIG
Unlock Script (scriptSig) is provided by the user to resolve the encumbrance
Lock Script (scriptPubKey) is found in a transaction output and is the encumbrance that must be fulfilled to spend the output

Figure 4.8: Pay-to-Public-Key-Hash (P2PKH) combining scriptSig and scriptPubKey [4]

Bitcoin standard transactions mostly use Pay-to-Public-Hash script (P2PKH Script), where the transactions process through the Bitcoin network. Bitcoin also has a script for Multi-signature, Pay-to-Script-Hash (P2SH), and pay-to-Public-Key.

The P2PKH Script is validated by running inside the Bitcoin network and is not shown to anyone, it only stores the hash of digital signature and hashes public key in the block explorer as Sigscript.

P2PKH Script

The notation of a P2PKH script is shown belown:

$< Sig >< PubKey > OP\_DUPOP\_HASH160 < PubkeyHash > OP\_EQUALVERIFYOP\_CHECKSIG$

A P2PKH script operates with 8 steps, as described below [4]:

1. The executions started from left to right.
2. The value of $< sig >$ is pushed to the top of the stack.
3. The value $Pubkey$ is pushed to the top of the stack on top of $< sig >$.
4. The dup operator duplicates the top item in the stack and pushes the result to the top of the stack.
5. The Hash160 operator hashes the top item in the stack with RIPEMD160(SHA-256(Pubkey)). The result value (PubKeyHash) is pushed to the top of the stack.
6. The value of $< PubKeyHash >$ is pushed on top of the value PubKeyHash calculated previously from the Hash160 of the Pubkey.
7. The EQUALVERIFY operator compares the PubKeyHash from the transaction with the PubKeyHash calculated from the user's PubKey. If they match, both are removed, and execution continues. Meaning they compare $PubKey$ and $Hash160$ if they are the same.
8. The CHECkSIG operator checks that signature $< sig >$ matches the public key $< PubKey >$ and pushes true to the top of the stack if true. The check between signature sig and public key $< pubkey >$ uses standard EDSA verification explained above in DSA signature verification.

4.4.4 Proposal for Post-quantum Blockchain

In this section, we propose a solution for a Bitcoin blockchain base on the public keys with PQ-OTS (post-quantum one-time signature). On the left side of the figures, 4.9 below shows the steps required for creating a standard Bitcoin address, and the right side shows the corresponding steps required for creating our proposed post-quantum Bitcoin address.

We want to add PQ-OTS to make the public key and Bitcoin address more robust against potential attacks from a powerful quantum computer.

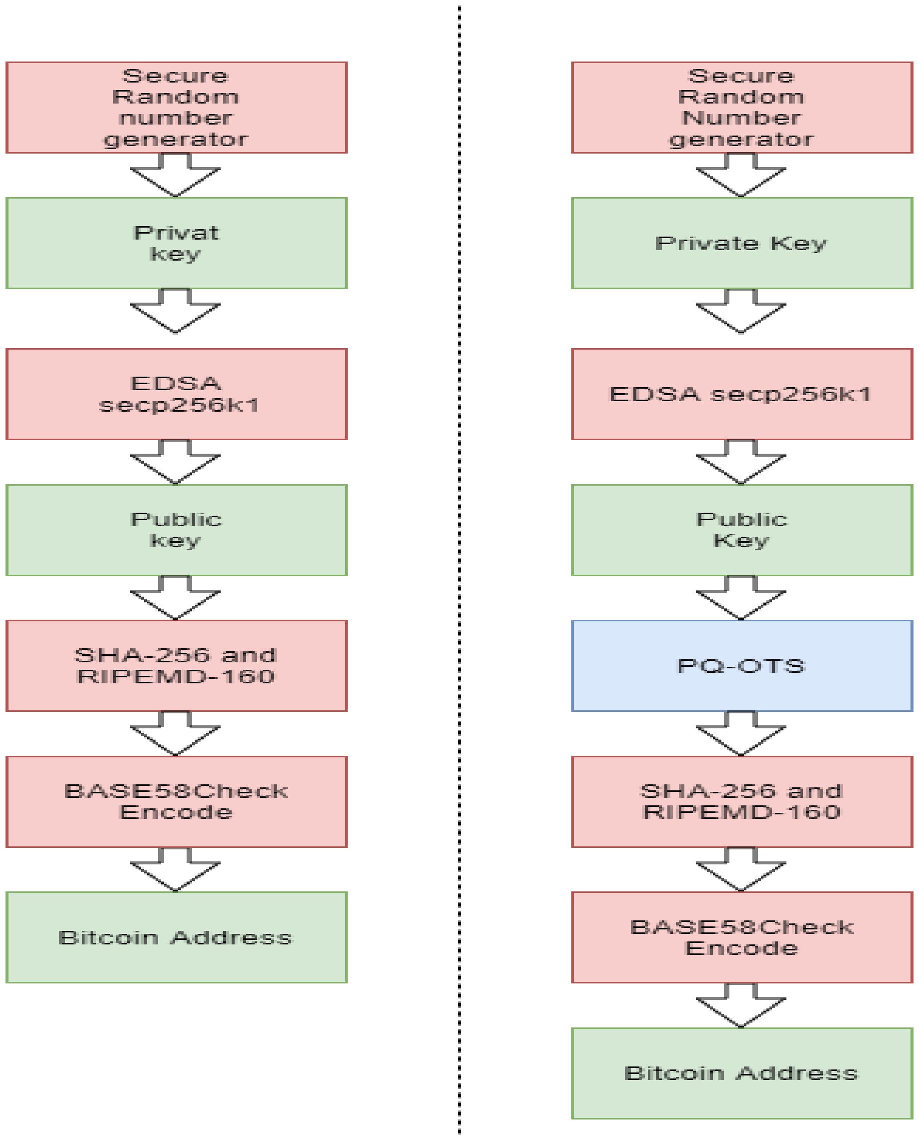


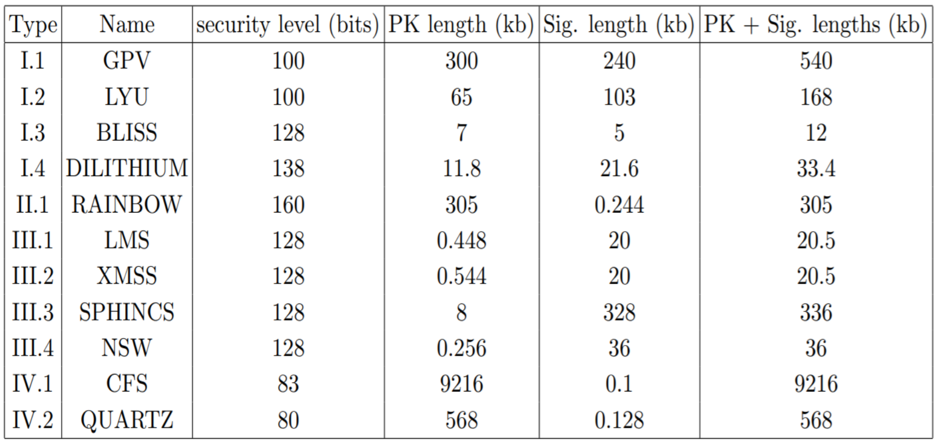
Figure 4.9: Traditional Bitcoin address (left) and PQ Bitcoin address (right).

Post-Quantum one-time Signature for Making a Bitcoin Address

Post-quantum one time signature (PQ-OTS) can be any post-quantum signature that is recommended by the National Institute of Standards and Technology (NIST) and Internet Engineering Task Force (IETF). The PQ-OTS will protect the public key from being known by anyone, and only the Bitcoin address will be shown in the public network.

We believe that replacing ECDSA cryptography is not necessary because it is a simple and easy way to create a private key and public key. ECDSA Verification is simple and does not take much computational power. We only need to be concern about protecting what ECDSA cryptography has created by adding a more robust cryptography mechanism against a quantum computer. This is similar to how we might need to protect the transmission data against potential attacks from powerful quantum computers in the future.

Below is figure 4.10 of a table that compares post-quantum cryptography key lengths. It can be seen that post-quantum cryptography mostly has larger key and signature sizes, and require more operations and memory that will create a constraint on the Bitcoin system. The block size of Bitcoin maybe needs to increase from 1MB to 2MB to handle increasing operations and key length, block time, and mining difficulty also needs to change.



Type	Name	security level (bits)	PK length (kb)	Sig. length (kb)	PK + Sig. lengths (kb)
I.1	GPV	100	300	240	540
I.2	LYU	100	65	103	168
I.3	BLISS	128	7	5	12
I.4	DILITHIUM	138	11.8	21.6	33.4
II.1	RAINBOW	160	305	0.244	305
III.1	LMS	128	0.448	20	20.5
III.2	XMSS	128	0.544	20	20.5
III.3	SPHINCS	128	8	328	336
III.4	NSW	128	0.256	36	36
IV.1	CFS	83	9216	0.1	9216
IV.2	QUARTZ	80	568	0.128	568

Figure 4.10: Post-quantum table of different public and signatures keys [24]

First proposal solution: Concatenation of ECDSA and PQ-OTS

We want to concatenate the ECDSA public key with PQ-OTS.

Process for adding PQ-OTS:

1. Generate ECDSA private key and ECDSA Public key.
2. Generate the PQ-OTS key.
3. Concatenated ECDSA public key with the PQ-OTS key.
4. Hash with SHA-256 and RIPEMD-160.
5. The result will be a hashed public key hashed with the same size as the old Bitcoin hashed public key. SHA-256 gives us 32 bytes, and RIPEMD-160 gives us 20 bytes.

Now we have a public key hashed with PQ-OTS. We verify the public key hash using the same method as P2PKH Script. We need to add PQ-OTS to the script.

The script look like this in theory: *<sig> is the digital signature, <PubKey> is the ECDSA public key, <PQ_OTS> is the quantum-safe signature.*

$< sig >< PubKey >< PQ\_OTS >$ OP_DUP OP_CAT OP_HASH160 $< PubkeyHash >$ OP_EQUALVERIFY OP_CHECKSIG

1. The executions goes from left to right.
2. The value of $< sig >$ is pushed to the top of the stack.
3. The Value *Pubkey* is pushed to the top of the stack, on top of $< sig >$.
4. Value of $< PQ\_OTS >$ is pushed to the top of the stack, on top of $< Pubkey >$.
5. The dup operator duplicates the top item in the stack and pushes the result to the top of the stack.
6. The Cat operator concatenates two strings, *Pubkey* and *PQ_OTS*. NB: As for now, the operator Cat is not valid in the Bitcoin script [25] because it could cause memory overflow with many DUP operators.
7. Hash160 operator hashes the top item in the stack with RIPEMD160(SHA-256(Pubkey+PQ_OTS)). The result value (PubKeyHash) is pushed to the top of the stack.
8. The value $< PubKeyHash >$ is pushed on top of the value PubKeyHash calculated previously from the Hash160 of the Pubkey.
9. The EQUALVERIFY operator compares the PubKeyHash from the transaction with the PubKeyHash calculated from the user's PubKey. If they match, both are removed, and execution continues. Meaning they compare *PubKey*, and $Hash160$ (PubKeyHash) if they are the same.
10. The CHECKSIG operator checks that signature $< sig >$ matches the public key $< PubKey >$ and pushes true to the top of the stack if true. The check between signature *sig* and public key $< pubkey >$ uses standard ECDSA verification explained above in DSA signature verification.

From our point of view, ECDSA cryptography used in Bitcoin is not at risk from a quantum computer, is because it has the protection of hash function SHA-256 and RIPEMD-160. The hash functions are our first line in defense against attackers by quantum computers. Grover's algorithm speed-up calculation to reduce the time needed for breaking hash functions by a factor of 2 or $\sqrt{2}$. Instead of calculating 256-bits or 2^{256} of numbers in SHA-256, Grover's algorithm calculate 128 bits or 2^{128} of number. RIPEMD-160 operates in 160-bits. The complexity of attacks using Grover's algorithm on the RIPEMD-160 be at 80 bits.

Security Analysis of the Proposed PQ Blockchain:

The transaction script has a potential vulnerability against attacker intercept when the transaction script scriptsig (digital signature and public key) is running in memory and writing data to the stack. The adversary/ attacker could try to steal information from scriptsig and running it with Grover's algorithm to find the private key. The more advanced way report in this paper [83]; The attacker tries to exploit the script program language writing to stack by using script command [25] "OP_DROP" to replace with another command in the stack since the scripts are running in close environment attacker needs to get "CHECKSIG" script to verify, and the transaction hash is written to the block than is too late to do anything about it.

Second, our PQ-OTS is a one-time signature or a few-time signature. PQ-OTS signature is new and is based on theory. PQ-OTS may create more vulnerabilities for the Bitcoin network and expose the Bitcoin network for common attacks and needs to be carefully supervised and tested.

We also need a safe place to generated these quantum-safe signatures. We have two possible choices either let owners of the private key generated the PQ-OTS key from their wallet or Bitcoin network created a new database for adding PQ-OTS key to the ECDSA public key. How do we keep our PQ-OTS from being used more than once, even for a few-time signature? In our proposed scheme, a Bitcoin user will use it twice.

Second proposed solution: Iteration of the hash

Bitcoin ECDSA public key only shows once, which is when the transaction verification and authentication happens inside a script. Bitcoin developers have stated that a Bitcoin address is unsafe after being used more than once or twice, because the possibility that an adversary could find the private key will increase after each time it is used. In addition, a Bitcoin private key is safe from a powerful quantum computer as long as the public key is not used. What if we could make the Bitcoin Address change after being used once or twice? This will solve the Bitcoin problem of reusing the same Bitcoin address.

In our second proposal PQ-OTS, we want to use a part of the Wintersnitz one time signature scheme (W-OTS) variant from the 2011 iterations idea. The W-OTS protocol is used in the hash-based signature scheme SPHINCS.

We only use a part of the W-OTS protocol: The pseudo-random generator that creates the private key is not needed, since we use the ECDSA public key to be our private key.

Process adding 256 random iteration hash public key :

1. Created an ECDSA private key.

2. Hash the private key 256 times with SHA2-256, where each hash is a possible hash of the public key.

3. Choose at random one of the hashes to be hashed with SHA-256 and RIPEMD-160.

4. Apply Based58Check Encode, which produces the Bitcoin address.

We have 256 hashes, where each can be used for the public-key (PK), each of the PK has an index n, where n can be 1,2,3...,256, Any of these public keys is denoted PK^n. The PK^n will be a random choice algorithm.

The output hash changes completely for the smallest change in the input. Hence, if we change the PK^n number, the Bitcoin address changes completely. Now we have 256 public hashed Bitcoin addresses that can be used. We do not need to reuse the Bitcoin address than necessary.

For verification and authentication of the 256 public key hashed, to prove that one of the 256 hash belongs to the ECDSA private key. We send total times n that has been hashed the ECDSA public key, and we will arrive at the number PK is hashed.

For verification between the ECDSA public key and ECDSA private key will happens inside the P2PKH script.

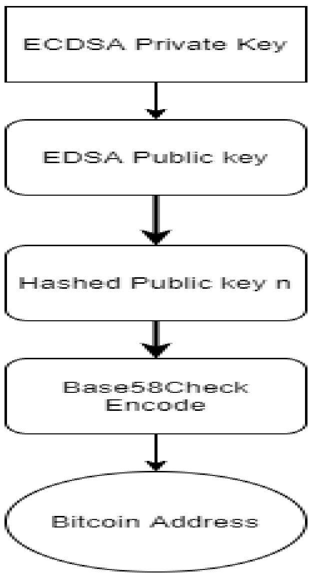


Figure 4.11: Iteration public key

The script can e.g. look like this;

$< sig >< PubKey >< N >$ OP_2DUP OP_SHA256N OP_HASH160 $< PubkeyHash >$ OP_EQUALVERIFY OP_CHECKSIG

1. The executions goes from left to right.

2. The value of $< sig >$ is pushed to the top of the stack.

3. The value $Pubkey$ is pushed to the top of the stack, on top of $< sig >$.

4. Value of $< \ N \ >$ is pushed to the top of the stack, on top of $< Pubkey >$.

5. The 2Dup operator duplicates the 2 top items in the stack and pushes the result to the top of the stack.

6. SHA-256N takes $< Pubkey >$ values and hashes $< N >$ times.. The result on the stack will be <PubHashN>. Also Remove duplicated <N> and <Pubkey> The SHA-256N function is not implement in Bitcoin script yet.

7. Hash160 operator hashes the top item in the stack with RIPEMD-160 (SHA-256 (pubKeyHashN)).

   The result value Hash160(PubKeyHashN) will be PubkeyHashA and is pushed to the top of the stack. Remove <PubHashN>.

8. The value $< PubKeyHash >$ is pushed on top of the value Hash160 (PubKeyHash).

9. The EQUALVERIFY operator compares the PubKeyHash from the transaction with the PubKeyHash calculated from the user's $< PubKey >$. If they match, both are removed, and execution continues. Meaning they compare PubKeyHash and $Hash160$ (PubKeyHash) if they are the same.

10. The CHECKSIG operator checks that signature $< sig >$ matches the public key $< PubKey >$ and pushes true to the top of the stack, if true.

    The check between signature sig and public key $< \ pubkey \ >$ uses standard ECDSA verification explained in section 3.4.2 Digital signatures Algorithm.

In theory the script on stack will looks like this;

$< sig >< PubKey >< N >$ OP_2DUP OP_SHA256N OP_HASH160 $< pubkeyHash >$ OP_EQUALVERIFY OP_CHECKSIG

Stack	Script	Description
Empty	Reading script command	scriptSig and scriptPubKey are combined.
<sig>	<sig>	Constants signature sent to stack.
<sig> <pubkey>	<Pubkey>	Constants public key add to stack op top of the signature.
<sig> <pubkey> <N>	<N>	The number of hashes is added to stack on top of the public key.
<sig> <pubkey> <N> <pubkey> <N>	OP_2DUP	Duplicated 2 top items in the stack, the public key and number of hashes.
<sig> <pubkey> <N> <PubHashN>	OP_SHA-256N	Hash Public key (pubkey) N times.
<sig> <pubkey> <PubHashA>	OP_HASH160	Hash top item with RIPEMD160(SHA-256(PubHashN)). <N> <PubHashN> is removed from the stack when done. Operation continues
<sig> <pubkey> <PubHashA> <pubKeyHash>	<pubKeyHash>	Add the public key hash from the script that creates the Bitcoin address for the users.
<sig> <pubkey>	OP_EQUALVERIFY	Compare <PubHashA> and <pubkeyHash> If they match. Remove <PubHashA> and <pubkeyHash>, operation continues.
<sig> <pubkey>	OP_CHECKSIG	Checks that the signature matches the EDSA public key. If true, push to stack.
true	Empty	If true transactions are valid, else execution fails.

Security Analysis of the second proposal for Bitcoin:

- The public key is visible in the script and to the network, which is the same problem as in our first proposal.
- In practice, users can use the same Bitcoin address as many times as they want, where the exposure to cryptanalysis increases each time a key is used. Almost all the user in Bitcoin use their Bitcoin address more than two times; they use for payment, for exchange, and for collecting rewards of mining. Changing the Bitcoin address too often will make many users confused, so that they, by accident sending their coins to the wrong address, and sometimes they forget that their address has been changed and share their old address to someone they knew.

4.4.5 Reason for Not Replacing the ECDSA Digital Signature

It has been predicted that ECDSA will be broken by quantum computers around the year 2027 [24]. Details on how a quantum implementation of Shor's algorithm potentially can be used to break ECDSA can be broken described [49]; *A 160-bit elliptic curve cryptographic key could be broken on a quantum computer using around 1000 qubits while factoring the security-wise equivalent 1024 bit RSA modulus would require about 2000 qubits*.

Bitcoin and many blockchain systems are decentralized systems. It is theoretically more comfortable to change a digital signature on a centralized system than a decentralized system. In a decentralized system, a consensus is needed to be agreed on before it can be modified, all or over 50% of the nodes must agree on updates and vote to decide new rules, and the old rules have to vanish, no one can be using the old signature schemes because they might be vulnerable to quantum attacks.

The old ledger or the old block that connects with each other will also be abandon since they contain and are build on digital signature ECDSA, an attacker can use the old data block to find out the private key even if a user now uses post-quantum signatures.

Bitcoin needs to move away from the old ledger and create a whole new genesis block, which in practice results in the creation of a new form of Bitcoin. This will cause some problem for Bitcoin:

1. The price of Bitcoin will fall, no one is going to keep their Bitcoin if they think Bitcoin will be abandon.
2. A hard-fork or soft-fork (upgrade) from signatures ECDSA to post-quantum signatures might reduce the value of funds linked to ECDSA addresses. Bitcoin users all have a private key that no one can know about, only the owners of that private key knows. The transition from signature ECDSA to quantum-safe all users needs to make new private key, new address, and move the funds to it. If some

do not make it in time before the new upgrades and new genesis block are created, their funds will be lost. The ECDSA private key can not be used to verified ownership in the new upgrades since the block does not exist anymore. A solution will be that Bitcoin can create a swap mechanism between the Bitcoin and the new upgrades. Mitigation from an old Bitcoin address will take around six months before it can be used and secure recommendations by this paper [40]. For those that have held Bitcoin in the old Bitcoin ledger for years and have not followed news and updates, the result might be loss of funds. Because at some points in the future, the chain needs to be stable and want to know how much the total circulating supply on the network, there will be a time-limited for a swap. Not to mention that the old block is vulnerable to a quantum attack so that there is a risk of their coins being stolen.

3. Mining is the work-force in Bitcoin, where miners use specialized mining hardware equipment ASICs for solving the Bitcoin puzzle faster to earn rewards. ASICs mining gear will maybe be useless with the new quantum-safe signature. The Bitcoin puzzle is based on finding a hash with specific properties. With PQ-DSA addresses, it might be necessary to use new algorithms and new equipment for mining. Many Miners will not accept these changes to the new quantum-safe signature, and the consensus will never be reached among the miners. Even if the consensus is reached, many will likely abandon Bitcoin and start mining other profitable ASICs coins. Without mining to help out, Bitcoin network security will be decreased and are vulnerable for many other standard attacks than just quantum attacks.

4. The block size in Bitcoin is 1MB. A block contains data such as transaction data, previous hash, etc., 1MB can have a maximum of 4000-4424 transactions, average 3.3 to 7 transactions per second. Mitigation to quantum-safe, block size will have to increased 1MB to xMB, and also, the average 10 min block time has to be considered changing since many quantum-safe algorithms have a bigger signature key than the ECDSA signature. Bigger block size and block time will expose more time for an attack such as DDOS, Elipse, etc. to happens before the block is signed and added to the ledger. The discussion of Bitcoin to have a bigger block size is still an ongoing topic [11].

5. Exchange and the third party. Almost all the quantum-safe signature algorithms generate one-time signatures that can only be used once and can never reuse. There is some few-time signature algorithm that lets us use more than one time, such as SPHINCS+ which are also hash-based. Not being able to reuse an address more than one time is a big problem that needs to change. The IOTA blockchain uses W-OTS [41] to generate a digital signature, and it has been reported that some users lose funds by reusing their address. This article [90] talks about the vulnerabilities of IOTA digital signature W-OTS.

In theory, exchange and each transaction with a third party will require that the users address to change. The user uses the exchange to buy and sell coins for each transaction that happens; a new signature or address has to be created. Assume that the average trading volume is 50 million US dollars per day. Will there be any exchange that has a system strong enough to handle the change of signature each time a transaction is going through. Exchange and a third party may need to convert the quantum-safe signature to something that will enable addresses to be used more than once. If an incident should happen, responsible lie with the exchange and third party.

4.5 Conclusion

In the beginning of this master's project, we intended to find a quantum-safe signature algorithm for the blockchain cryptocurrency Bitcoin, and have a look at some use cases of Blockchain technology. For this purpose, we have studied and analyzed Bitcoin, blockchain systems, and quantum-safe signatures. We find that it is difficult to replace the ECDSA digital signature with a quantum-safe signature for Bitcoin. Bitcoin could lose support from users, miners, and exchanges. The outcome might be more harmful than useful in the short term, but in the long term protecting Bitcoin chain and transactions is more important to make users feel safe against quantum attacks. Quantum computer development is going at a fast speed where big companies such as Google, IBM, and Honeywell are competing with each other to be the leader in the development, not to mention countries that develop this technology in secret. We may see a quantum computer that can break digital signature RSA and ECDSA before 2027.

We feel that we do not have enough understanding and knowledge to propose the perfect solutions for a post-quantum Bitcoin cryptocurrency. We have focused on solutions for protecting the ECDSA signature public-key signature. We have proposed two solutions. The first solution is based on concatenating the ECDSA public-key with quantum one-time signature keys. The second solution is based on iterating the ECDSA public key hashed many times, so the Bitcoin address not used more than 1-2 times.

The main problem with our two suggestions is that a transaction needs to be verified and authenticated by using the script language, and sending out our ECDSA signature and ECDSA public key to the network, will expose our keys for the common attack. We need to find a solution to encrypt our transaction signature between network nodes to become quantum-safe. TLS (Transport Layer Security) uses using Diffie Hellman key exchange and a trusted third-party authentication server. We could send our ECDSA signature and ECDSA public key to the trusted authentication server for authentication, so when we want to verify the ownership of an address, the trusted authentication server could handle it without giving out the ECDSA public key. Then Bitcoin will not be a 100% decentralized system because it needs a trusted authentication server to verify. Bitcoin systems are built on that they do not need to trust any nodes or any external system for verified, but based on the previous hash block of transactions and any user can verify their ownership of an address themselves. Users can have 100% ownership of their address and private key. Bitcoin in the future may become worthless if they can not find a way to solve the threats from quantum attacks.

A governing cryptocurrency system may solve some of the problems Bitcoin and other crypto-currencies will face against a quantum computer. Sweden is working on e-krone cryptocurrency. Japan is working on the MUFG coin that is said to be released in 2020 or 2021. China is working on BSN Public Chain Integration Plan with six blockchains: Nervos, Neo, Tezos, Irisnet, Ethereum, and EOSIO for permissioned services, permissionless services, and interchain services. For the stability of these e-currencies, it is essential that they are designed to be resistant against attacks from future powerful quantum computers.

Bibliography

[1] Binance Academy. *Binance explain on sybil attack*. URL: https://www.binance.vision/security/sybil-attacks-explained.

[2] Andreas H¨ulsing. *W-OTS+– Shorter Signatures for Hash-Based Signature Schemes*. 2013. URL: https://huelsing.net/wordpress/wp-content/uploads/2013/05/wotsspr.pdf.

[3] Andreas Huelsing, Denis Butin, Stefan-Lukas Gazdag, Joost Rijneveld, Aziz Mohaisen. *XMSS: eXtended Merkle Signature Scheme*. May 2018. URL: https://tools.ietf.org/html/rfc8391.

[4] Andreas M. Antonopoulos. *Mastering Bitcoin*. July 2017. URL: https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch04.asciidoc#base58check_encoding.

[5] Andreas M. Antonopoulos. *Mastering Bitcoin*. Springer-Verlag New York Inc, 2017. ISBN: ISBN 0-387-95273. URL: https://github.com/bitcoinbook/bitcoinbook/blob/develop/ch04.asciidoc#ecc-curve.

[6] USENIX Multi author. *Eclipse Attacks on Bitcoin's Peer-to-Peer Network*. August 2015. URL: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-heilman.pdf.

[7] Adam Back. *Hash cash postage implementation*. 1997. URL: http://www.hashcash.org/papers/announce.txt.

[8] Adam Back. *Hashcash - A Denial of Service Counter-Measure*. 2007. URL: http://www.hashcash.org/papers/hashcash.pdf.

[9] ECB Bank. *Report on electronic money*. URL: https://www.ecb.europa.eu/pub/pdf/other/emoneyen.pdf. (accessed: 07.08.2019).

[10] ECB Bank. *What is Money*. URL: https://www.ecb.europa.eu/explainers/tell-me-more/html/what_is_money.en.html. (accessed: 01.09.2019).

[11] Bitcoin wiki. *Block size limit controversy*. URL: https://en.bitcoin.it/wiki/Block_size_limit_controversy.

[12] Bitcoin.org. *Bitcoin Wallet ledger for install*. URL: https://bitcoin.org/en/choose-your-wallet?step=5.

[13] Bitcoin.org. *Block-height-and-forking*. URL: https://developer.bitcoin.org/devguide/block_chain.html.

[14] Blockchain.com. *Average Number Of Transactions Per Block*. URL: https://www.blockchain.com/charts/n-transactions-per-block. accessed: 05.09.2019.

[15] Blockchain.com. *Bitcoin first transaction to hal finney*. URL: https://www.blockchain.com/btc/tx/f4184fc596403b9d638783cf57adfe4c%2075c605f6356fbc91338530e9831e9e16. accessed: 01.04.2019.

[16] Bill Buchanan. *Quantum Robust: Winternits one time signature scheme(W-OTS)*. URL: https://asecuritysite.com/encryption/wint.

[17] Cardano.org. *Cardano Ada consensus Ouroboros*. 2017. URL: https://www.cardano.org/en/ouroboros/. accessed: 09.10.2019.

[18] PEW Research Center. *Mobile Fact Sheet*. URL: https://www.pewinternet.org/fact-sheet/mobile/.

[19] David Chaum. *Blind signatures for untraceble payment*. URL: http://blog.koehntopp.de/uploads/Chaum.BlindSigForPayment.1982.PD. (accessed: 07.09.2019).

[20] DR.Lily Chen. *Cryptographic Standards and Guidelines*. URL: https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-crypto-projects/aes-development.

[21] *Coinmarketcap.com*. URL: https://coinmarketcap.com/.

[22] Wei Dai. *B-Money*. 1998. URL: http://www.weidai.com/bmoney.txt. accessed: 05.02.2019.

[23] Daniel J. Bernstein1, Daira Hopwood, Andreas Hülsing, Tanja Lange3, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. *SPHINCS: practical stateless hash-based signatures*. 2014. URL: https://eprint.iacr.org/2014/795.pdf.

[24] Divesh Aggarwal, Gavin K. Brennen, Troy Lee, Miklos Santha, Marco Tomamichel. *Quantum attacks on Bitcoin, and how to protect against them*. 28 Oct 2017. URL: https://arxiv.org/abs/1710.10377.

[25] Bitcoin core documentation. *Script*. URL: https://en.bitcoin.it/wiki/Script.

[26] Microsoft Research John R. Douceur. *Sybil Attack*. August 2015. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2002/01/IPTPS2002.pdf.

[27] E.O. Kiktenko, A.A. Bulychev, P.A. Karagodin, N.O. Pozhar, M.N. Anufriev, A.K. Fedorov. *Sphincs+ digital signature scheme with Streebog hash function*. April 2019. URL: https://www.groundai.com/project/sphincs-digital-signature-scheme-with-gost-hash-functions/1#S1.F1.

[28] E.O. Kiktenko, N.O. Pozhar, M.N. Anufriev, A.S. Trushechkin, R.R. Yunusov, Y.V. Kurochkin, A.I. Lvovsky, A.K. Fedorov. *Quantum-secured blockchain*. 29 May 2017. URL: https://arxiv.org/abs/1705.09258.

[29] Bank of England. *How is money created*. URL: https://www.bankofengland.co.uk/knowledgebank/how-is-money-created. (accessed: 05.08.2019).

[30] Hal Finney. *Reusable Proofs of Work*. URL: https://nakamotoinstitute.org/finney/rpow/index.html. accessed: 01.07.2019.

[31] Bitcoingold forum. *Double Spend Attacks on Exchanges*. URL: https://forum.bitcoingold.org/t/double-spend-attacks-on-exchanges/1362.

[32] ELECTRONIC FRONTIER FOUNDATION. *EFF DES CRACKER MACHINE BRINGS HONESTY TO CRYPTO DEBATE*. URL: https://www.eff.org/press/releases/eff-des-cracker-machine-brings-honesty-crypto-debate.

[33] Joakim von zur Gathen. *CryptoSchool*. Springer verlag Berlin Heiderberg: Springer, 2015.

[34] Ilya Grigorik. *High Performance Browser Networking*. 1005 Gravenstein Highway North Sebastopol, CA 95472: O'Reilly Media, 2013. URL: %7Bhttps://hpbn.co/transport-layer-security-tls/%7D.

[35] Cointelegraph.com SAMUEL HAIG. *John McAfee Is 99% Certain He Knows Who Satoshi Nakamoto Is*. URL: https://cointelegraph.com/news/john-mcafee-knows-who-satoshi-nakamoto-is.

[36] Jonathan Hui. *Quantum Computing Series*. URL: https://medium.com/@jonathan_hui/qc-quantum-computing-series-10ddd7977abd.

[37] Hyperledger-fabric.io. *What is hyperledger frabric*. URL: https://hyperledger-fabric.readthedocs.io/en/latest/whatis.html. accessed: 10.12.2019.

[38] Hyperledger.org. *Hyperledger frabric*. URL: https://wiki.hyperledger.org/display/fabric/Hyperledger+Fabric. accessed: 10.12.2019.

[39] Hyperledger.org. *The Hyperledger Greenhouse*. URL: https://www.hyperledger.org/. accessed: 09.10.2019.

[40] I.Stewart, D.IIie, A.Zamyatin, S.Werner, M.F.Torshizi, W.J.Knottenbelt. *Committing to quantum resistance: a slow defence for Bitcoin against a fast quantum computing attack*. 1june 2018. URL: https://royalsocietypublishing.org/doi/10.1098/rsos.180410.

[41] IOTA Documentation. *Signature*. URL: https://docs.iota.org/docs/getting-started/0.1/clients/signatures.

[42] Juan Benet IPF. *IPFS whitepaper*. URL: https://github.com/ipfs/ipfs/blob/master/papers/ipfs-cap2pfs/ipfs-p2p-file-system.pdf. accessed: 09.12.2019.

[43] IPFS.io. *Content addressing*. URL: https://docs-beta.ipfs.io/concepts/what-is-ipfs/#content-addressing.

[44] IPFS.io. *What is IPFS*. URL: https://docs-beta.ipfs.io/concepts/what-is-ipfs/#decentralization. accessed: 09.12.2019.

[45] Jean-Philippe Aumasson, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe. *SPHINCS+ Submission to the NIST post-quantum project*. May 2019. URL: https://sphincs.org/data/sphincs+-round2-specification.pdf.

[46] Jehyuk Jang, Heung-No Lee. *Profitable Double-Spending Attacks*. 2018. URL: https://arxiv.org/ftp/arxiv/papers/1903/1903.01711.pdf.

[47] Johannes Buchmann, Erik Dahmen, Sarah Ereth, Andreas H¨ulsing, and Markus R¨uckert. *On the Security of the Winternitz One-Time Signature Scheme*. 2011. URL: https://eprint.iacr.org/2011/191.pdf.

[48] Johannes Buchmann, Luis Carlos Coronado Garc´ıa2, Erik Dahmen1, Martin D¨oring, and Elena Klintsevich. *CMSS – An Improved Merkle Signature Scheme*. 2006. URL: https://eprint.iacr.org/2006/320.pdf.

[49] John Proos, Christof Zalka. *Shor's discrete logarithm quantum algorithm for elliptic curves*. URL: https://arxiv.org/abs/quant-ph/0301141.

[50] Jonathan Katz. *Digital Signatures*. Milton keynes UK: Spring, 2010.

[51] Tanai Khiaonarong and David Humphrey. *Cash Use Across Countries and the Demand for Central Bank Digital Currency*. URL: https://www.imf.org/~/media/Files/Publications/WP/2019/WPIEA2019046.ashx. accessed: 17.09.2019.

[52] Anastasios Kyrillidis. *Introduction to quantum computing: Bloch sphere*. URL: http://akyrillidis.github.io/notes/quant_post_7.

[53] Leslie Lamport, Robert Shostak, Marshall Pease. *The Byzantine Generals Problem*. July 1982. URL: https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf. accessed: 09.09.2019.

[54] Libra.org. *Libra Protocol*. URL: https://developers.libra.org/docs/libra-protocol. accessed: 10.12.2019.

[55] Libra.org. *Libra whitepaper Introduction*. URL: https://libra.org/en-US/white-paper/#introduction. accessed: 10.12.2019.

[56] Statista.com Shanhong Liu. *Size of the Bitcoin blockchain from 2010 to 2019, by quarter*. URL: https://www.statista.com/statistics/647523/worldwide-bitcoin-blockchain-size/.

[57] Marco Lucamarini. *Viewpoint: Record Distance for Quantum Cryptography*. URL: https://physics.aps.org/articles/v11/111.

[58] Matt Lepinski, Sean Turner. *BGPsec Protocol Specification*. URL: https://tools.ietf.org/html/draft-ietf-sidr-bgpsec-overview-08.

[59] Peter Maymounkov and David Mazieres. *Kademlia: A Peer to Peer Information system based on the XOR Metric*. URL: https://pdos.csail.mit.edu/~petar/papers/maymounkov-kademlia-lncs.pdf. accessed: 10.12.2019.

[60] Ralph C. Merkle. *A Digital Signature Based on a Conventional Encryption Function*. 1987. URL: https://people.eecs.berkeley.edu/~raluca/cs261-f15/readings/merkle.pdf. accessed: 05.09.2019.

[61] Ricahard A. Mollin. *RSA and Public-key Cryptography*. Boca Raton, Florida 33431: CRC press LLC, 2003.

[62] Trygve Larsen Morset. *Forelesning penger,inflasjon og finanspolitikk*. URL: https://www.uio.no/studier/emner/sv/oekonomi/ECON1310/v17/forelesning10pengerv2017.pdf. (accessed: 05.08.2019).

[63] Andysah Putera Utama Siahaan Muhammad Iqbal. *Combination of MD5 and ElGamal in Verifying File Authenticity and Improving Data Security*. URL: https://www.researchgate.net/publication/328693265_Combination_of_MD5_and_ElGamal_in_Verifying_File_Authenticity_and_Improving_Data_Security.

[64] Computer history museum. *A brief History*. URL: https://www.computerhistory.org/babbage/history/.

[65] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: https://bitcoin.org/bitcoin.pdf. accessed: 01.05.2019.

[66] Tron network. *What is Tron*. URL: https://developers.tron.network/docs. accessed: 09.10.2019.

[67] Nick$_S$*zabo*. *Bit Gold*. 2005. URL: https://nakamotoinstitute.org/bit-gold/. accessed: 01.07.2019.

[68] NIST. *Post Quantum Cryptography*. URL: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-2-submissions.

[69] Peercoin.net. *Introduction to Peercoin*. 2012. URL: https://docs.peercoin.net/#/consensus-algorithm. accessed: 09.09.2019.

[70] Peercoin.net. *Peercoin consensus Proof of Stake*. 2012. URL: https://university.peercoin.net/#/9-peercoin-proof-of-stake-consensus. accessed: 09.09.2019.

[71] The New York Time Nathaniel Popper. *Decoding the enigma of satoshi nakamoto and the birth of bitcoin*. 2015. URL: https://www.nytimes.com/2015/05/17/business/decoding-the-enigma-of-satoshi-nakamoto-and-the-birth-of-bitcoin.html. accessed: 05.08.2019.

[72] Roy Arends, Rob Austein, Matt Larson, Dan Massey, Scott Rose. *DNS Security Introduction and Requirements*. URL: https://www.ietf.org/rfc/rfc4033.txt.

[73] Andrew S.TanenBaum and Maarten Van Steen. *Distributed Systems Principles and Paradigms*. Upper Saddle River, NJ07458: Pearson Education Inc, 2007.

[74] Kai Sedgwick. *Eight Historic Bitcoin Transactions*. URL: https://news.bitcoin.com/eight-historic-bitcoin-transactions/. accessed: 01.03.2019.

[75] National Institute of Standard. *Announcing Approval of Federal Information Processing Standard*. URL: https://csrc.nist.gov/news/2001/announcing-approval-of-fips-197-aes.

[76] National Institute of Standard. *Recommendation for Key Management*. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

[77] National Institute of Standards and Technology. *Digital Signature Standard (DSS) FIPS PUB 186-4*. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[78] Stephen Kent, Karen Seo. *Security Architecture for the Internet Protocol*. URL: https://tools.ietf.org/html/rfc4301.

[79] Emma Strubell. 'An Introduction to Quantum Algorithms'. In: *Quantum Tutorial* (Spring 2011). URL: https://people.cs.umass.edu/~strubell/doc/quantum_tutorial.pdf.

[80] Corda R3 team. *Blockchained Post-Quantum Signatures*. URL: https://github.com/corda/bpqs.

[81] Quantum Resistant ledger team. *Quantum Resistant ledger*. URL: https://theqrl.org/.

[82] Tezos.com. *What is Tezos, get started*. URL: https://tezos.com/get-started/. accessed: 09.10.2019.

[83] Ubaidullah, Fizza Abbas, Rasheed Hussain, Hasoo Eun, Heekuck Oh. *A Simple Yet Efficient Approach to Combat Transaction Malleability in Bitcoin*. 27 July 2015. URL: https://www.researchgate.net/publication/272969568_A_Simple_Yet_Efficient_Approach_to_Combat_Transaction_Malleability_in_Bitcoin.

[84] Bitcoin wiki. *Base58Check encoding*. URL: https://en.bitcoin.it/wiki/Base58Check_encoding.

[85] Bitcoin wiki. *SHA-256*. URL: https://en.bitcoinwiki.org/wiki/SHA-256.

[86] Wikipedia. *Grover's algorithm*. URL: https://en.wikipedia.org/wiki/Grover%5C%27s_algorithm.

[87] Wikipedia. *Quantum Logi gate*. URL: https://en.wikipedia.org/wiki/Quantum_logic_gate.

[88] Wikipedia. *Sphere*. URL: https://en.wikipedia.org/wiki/Sphere.

[89] Frank Wilczek. *Entanglement Made Simple*. URL: https://www.quantamagazine.org/entanglement-made-simple-20160428.

[90] Willem Pinckaers. *IOTA Signatures, Private Keys and Address Reuse?* URL: http://blog.lekkertech.net/blog/2018/03/07/iota-signatures/.

[91] Reuter.com Brett Wolf. *U.S. seeks forfeiture against digital-money firm*. URL: https://www.reuters.com/article/financial-egold-idUSN0712511420110607. accessed: 16.09.2019.

[92] Ronald de Wolf. 'Quantum Computing: Lecture Notes'. In: *arXiv:1907.09415v1 [quant-ph]* (19 Jul 2019). URL: https://homepages.cwi.nl/~rdewolf/qcnotes.pdf.