UNIVERSITY OF VAASA

FACULTY OF TECHNOLOGY

BLOCKCHAIN: ANALYSIS, COMPARISON AND CRITIQUES

Master's thesis for the degree of Master of Industrial Digitalisation submitted for assessment, Vaasa, September 2018

VAASA 2018

TABLE OF CONTENTS

page

TABLE OF FIGURES AND TABLES **4**

ABBREVIATIONS **6**

ABSTRACT: **9**

1. INTRODUCTION **11**

1.1. Motivation 11

1.2. Thesis Structure 12

1.3. Thesis Research Clarifications 12

2. BITCOIN **13**

2.1. Introduction 13

2.2. Why Bitcoin? 13

2.3. Bitcoin Network Overview 14

2.4. Users Addresses and Keys 16

2.4.1. Public Key Cryptography 16

2.4.2. Digital Signature 18

2.4.3. Bitcoin Addresses 19

2.5. Bitcoin Wallets 21

2.6. Bitcoin Transactions 25

2.6.1. Transaction Scripts 28

2.7. The Blockchain 30

2.7.1. Merkle Trees 32

2.8. Bitcoin Mining 34

2.8.1. Decentralized Consensus Protocol 35

2.8.2. Proof-of-Work Challenge 35

2.8.3. Blockchain Forks 37

2.9. Consensus Attack 40

2.10. Critiques 40

3. ETHEREUM **43**

3.1. Introduction 43
3.2. Ethereum Network Overview 44
3.3. Transactions 46
3.4. Ethereum Merkle Tree 48
3.5. Ethereum Client 49
3.6. Smart Contract 50
3.6.1. Solidity Programming Language 51
3.6.2. Tokens 53
3.7. The Anatomy of DApps 56
3.7.1. Truffle Suite 59
3.7.2. DApps Use Cases 60
3.8. Critiques 61

4. THE HYPERLEDGER PROJECT 65
4.1. Introduction 65
4.2. Hyperledger Sawtooth 68
4.2.1. Transactions 69
4.3. PoET Consensus 71
4.4. Validator's Journal 72
4.5. Network and Transactions Permissions 74
4.6. The Food Journey 74
4.7. Critiques 75

5. Conclusion 77

6. LIST OF REFERENCES 80

TABLE OF FIGURES AND TABLES

Figure 1. Bitcoin Network Overview 15
Figure 2. An Elliptic curve 18
Figure 3. Digital Signature in Public Key Cryptography 18
Figure 4. Generating Bitcoin Address 20
Figure 5. Bitcoin Paper Waller 21
Figure 6. List of Popular Wallets 22
Figure 7. Bitcoin Hardware Wallet 22
Figure 8. HD Deterministic Wallet 24
Figure 9. Transactions Overview 26
Figure 10. Transaction Data Structure 27
Figure 11. The Blockchain Ledger 30
Figure 12. Block Data Structure 31
Figure 13. Merkle Tree 32
Figure 14. Merkle Path 33
Figure 15. Bitcoin Energy Consumption Chart 34
Figure 16. Temporary Blockchain Fork 38
Figure 17. Temporary Blockchain Fork Resolved 39
Figure 18. The Structure of Ethereum Transaction 47
Figure 19. Ethereum Gas Station Website 47
Figure 20. Ethereum Full Node Client 50
Figure 21. Casino Tokens 54
Figure 22. Zon Startup Token 55
Figure 23. Web App vs DApp Core Components 56
Figure 24. Chunks Distribution Over Swam Nodes 58
Figure 25. Truffle Suite 60
Figure 26. CryptoKitties Dapp 60
Figure 27. DappRadar.com Website 61
Figure 28. Hyperledger Project Greenhouse 66
Figure 29. Hyperledger Sawtooth Network Overview 68
Figure 30. Sawtooth Transaction 70

Figure 31. Intel SGX Runtime Execution 72
Figure 32. Sawtooth Journal 73
Figure 33. Fish Journey from Ocean to Table 75

Table 1. Solving Hello, World! PoW 37

ABBREVIATIONS

API	Application Programming Interface
ASIC	Application-Specific Integrated Circuit
BaaS	Blockchain as a Service
BTC	Bitcoin Currency Unit
COBR	Concise Binary Object Representation
DAPPS	Decentralized Application
DAO	Decentralized Autonomous Organization
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DPA	Distributed Preimage Archive
ECDSA	Elliptic Curve Digital Signature Algorithm
ERC	Ethereum Request for Comments
ENS	Ethereum Name Service
EOA	Externally Owned Account
ETH	Ethererum Currency Unit
EVM	Ethereum Virtual Machine
FFG	Friendly Finality Gadget
HMAC	Hash-based Message Authentication Code
IBAN	International Bank Account Number
ICAP	Inter exchange Client Address Protocol
ICO	Initial Crowdfund Offering
ILP	Interledger Protocol
IPNS	InterPlanetary Naming Service
JSON	JavaScript Object Notation
LIFO	Last In First Out
LSB	Least Significant Byte
NIST	National Institute of Science and Technology
P2P	Peer to Peer
P2PKH	Pay-To-Public-Key-Hash

P2SH	Pay to Script Hash
PoET	Proof of Elapsed Time
PoS	Proof of Stake
PoW	Proof of Work
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
ROM	Read Only Memory
RPC	Remote Procedure Call
SDK	Software Development Kit
SGX	Software Guard Extensions
SHA	Secure Hash Algorithm
SPV	Simplified Payment Verification
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
UTXO	Unspent Transaction Outputs

UNIVERSITY OF VAASA
School of School of Technology and Innovations
Author: Mohammad Nour Saffaf
Topic of the thesis: Blockchain: Analysis, Comparison, and Critiques
Degree: Master of Science in Industrial Digitalisation
Master's Programme: Industrial Digitalisation
Supervisor: Professor Mohammed Elmusrati
Year of entering the University: 2016
Year of completing the thesis: 2018
Number of pages: 84

ABSTRACT:

Nowadays, we are witnessing a rapid development in newer technologies that might change our lives in future. Robotics, Artificial Intelligence, Internet of Things and Blockchain are part of the fourth industrial revolution. While it might be easier to imagine how AI, robotics and IoT are changing our future, blockchain is surrounded with hype and doubts. Billions of dollars flooding into blockchain projects, large cooperate are working on enterprise solutions, and countries considering blockchain digital currencies motivated by political reasons. On the other side, enthusiasts driven by the idea to change the financial system, anti-capitalist, criminals, and hackers found alternative in the blockchain. This thesis aims toward understanding blockchain technically and to highlight the main differences between the main three blockchain solutions available today. Understanding the concepts of the blockchain and trying to answer questions regarding the future of blockchain and how the technology can be utilized to solve problems other technologies have failed.

This master thesis explores this exciting technology of blockchain and analysis its strengths, challenges, opportunities, and future. Starting from Bitcoin, then going through Ethereum and finally toward Hyperledger enterprise solutions. The study presents technical details, programming concepts, usages, limitations and critiques.

Blockchain is a new technology and in few years, researches and experimental projects will reveal where the technology would stand. It might disturb fundamentally many industries or just return back to where it originated from, the Bitcoin.

KEY WORDS: bitcoin, blockchain, ethereum, hyperledger, sawtooth, ledger, cryptocurrency

1. INTRODUCTION

We are probably living in a period similar to the beginning of the 90s before the internet become widely used. Many people then have heard about the internet or used it shortly over slow dial-up connection. The internet has changed our lives in a way few people could have imagined then. The next revolution is near, where we are going to witness new emerging technologies that might change our lives in a way that only few people are able to imagine now. Robotics, artificial intelligence, machine learning, internet of things and blockchain are part of the what the World Economic Forum calls as *The Fourth Industrial Revolution*. Many people have seen self-driving cars and robotics on real life or in the media and what they offer for our future but the blockchain remains mystery. Despite the fact that blockchain is the heart of the Bitcoin digital currency, it can be used for other purposes.

1.1. Motivation

Technology keep evolving and changing and part of our responsibilities as master students studying the digitalization of the industry to keep an eye on future technologies. The blockchain technology might change fundamentally different industries in the future. Billions of dollars have already been invested by large companies and investors in blockchain. However, since the technology is still considered new and under rapid development, it lacks intensive technical documentations, books, open source projects and other resources that master student relies on during his studies. It has been challenging not only to me but to many others to understand what the Bitcoin is technically and how it works. It was difficult to understand where the blockchain is heading without looking deep. At this stage of blockchain, I believe it is essential to understand major blockchain solutions and what they are trying to achieve. This thesis can be considered a step toward being part of the future.

1.2. Thesis Structure

The blockchain technology is still at early stages therefore the thesis focuses on performing a technical analysis for the most promising solutions available today. The thesis starts with the Bitcoin since it where the blockchain innovation started. Then, the focus will move toward Ethereum and its big promises for new generation of the internet. Finally, the thesis discusses Hyperledger project which is considered as the most promising blockchain solution for the enterprise.

1.3. Thesis Research Clarifications

Understanding the details of different blockchain technologies at this time has been challenging due to lack of books and constant changes of the technical documentations. I was forced to rely on limited resources and sometimes incomplete books that have been scheduled for release in the next few months. By the time this thesis is published, changes may occur to the references. Blockchain is still evolving and most of the projects are experimental. I hope this thesis will serve as a quick guide for follow students interested in further researching in the blockchain technology.

2. BITCOIN

2.1. Introduction

On October ten years ago, a revolutionary research paper was published by unknown author who used a pseudonym name Satoshi Nakamoto with the title of “Bitcoin: A Peer-to-Peer Electronic Cash System”. Since then, Bitcoin technologies has caused disruption not only to the financial sectors but also to many fields including the industry and information technology.

Bitcoin is generally known as an electronic currency that can be used online without relying on banks or governments or any third party to authorize or facilitate the payments. The electronic currency runs on own network using technical protocols and solutions. Bitcoin network is known as *decentralized permissionless system*. Because there is no central control over the Bitcoin network, anyone can participate including users who want to buy goods, merchants who want to sell and exchange services that exchange fiat money with bitcoin currency (Rosenbaum 1-4).

From the technical point of view, bitcoin network uses technical concepts built together to create a monetary network. Some of these concepts (i.e. peer to peer protocol) were older inventions while others were new inventions by Satoshi Nakamoto. These new inventions were behind the technical and industrial disruption that resulted in billions of dollars in research and investments under the name of the blockchain technology.

2.2. Why Bitcoin?

Bitcoin offers secure, fast, borderless and most importantly independent monetary system. The current financial monetary system and their financial crises have convinced many people that an alternative system is required. At this time, it is estimated that only 62% of the world population own a bank account. Banks may deny their services for people for many reasons, such as ethnicity, no valid ID and even for political reasons.

Another problem is privacy where banks and governments track every single payment and every balance for all the population. This might sound like a weird problem for some people who believe that this is necessary for fighting crimes, but it has been misused by some countries around the world. For example, seizing bank accounts with large balances for financial rescue programs similar to what happened in Cyprus in 2013. A similar problem happens nowadays in Libya where banks have frozen people accounts due to a national political crisis. Bitcoin offers the possibility for anyone with internet connection to have her own account. Bitcoin also offers private accounting where only the user is aware about how much she owns and only she can control it (Rosenbaum 12-14).

Inflation is one of the biggest problem of the current financial system where prices of goods increase rapidly turning sometimes people savings into virtually zero similar to what happened recently in Venezuela where the local currency inflated 4000% in 2017 turning large population into poverty and hunger. Inflation occurs when governments misuse the monetary system by printing more money as a tool to cover national debut through extracting value from people. Another serious problem is transferring money between countries where the central banks control the decision to allow or to forbid the payments. This opened the door to third party companies or even individuals to facilities the payments with large fees. Bitcoin prevents high inflation by limiting the supply to 21 million BTC. The 21 million BTC are not available nowadays but *mined* in fixed rate until the year 2140 where the last bitcoin will be mined (fiat money is printed, bitcoin are mined as if it is gold). At the moment, the supply is about 17 million BTC and the mining rate is halving every four years (Rosenbaum 13-19).

2.3. Bitcoin Network Overview

Peer to peer network consists of computers known as nodes connected together using a software such as BitTorrent that makes it possible to share information without the need for a central server. Bitcoin network illustrated in Figure 1 consists of nodes connected together that run special bitcoin client software to create bitcoin network.

Regular users who want to purchase goods or transfer money use a *wallet*. Wallet are the gateway to the bitcoin network. However, wallets do not store bitcoin but only the encrypted keys and the address of the owner. Using a wallet, a user can construct a payment *transaction*. Transactions propagate through the network *nodes*. Network nodes are responsible for receiving, validating and forwarding transactions as well as validating and updating own copy of the database known as the *blockchain*. The blockchain is a transactions ledger that stores transactions in groups called *blocks* connected together to form a *chain* and this is how the name blockchain was created. The blockchain is immutable for editing, and therefore transactions cannot be deleted or altered. Each node holds own copy of the blockchain that can be validated for integrity with other nodes of the network through technique known as *consensus*. Some of the network nodes are also *miners*. Miners earn bitcoin by competing against each other on a difficult task to earn the right to publish the next block to the blockchain (Rosenbaum 4-12).

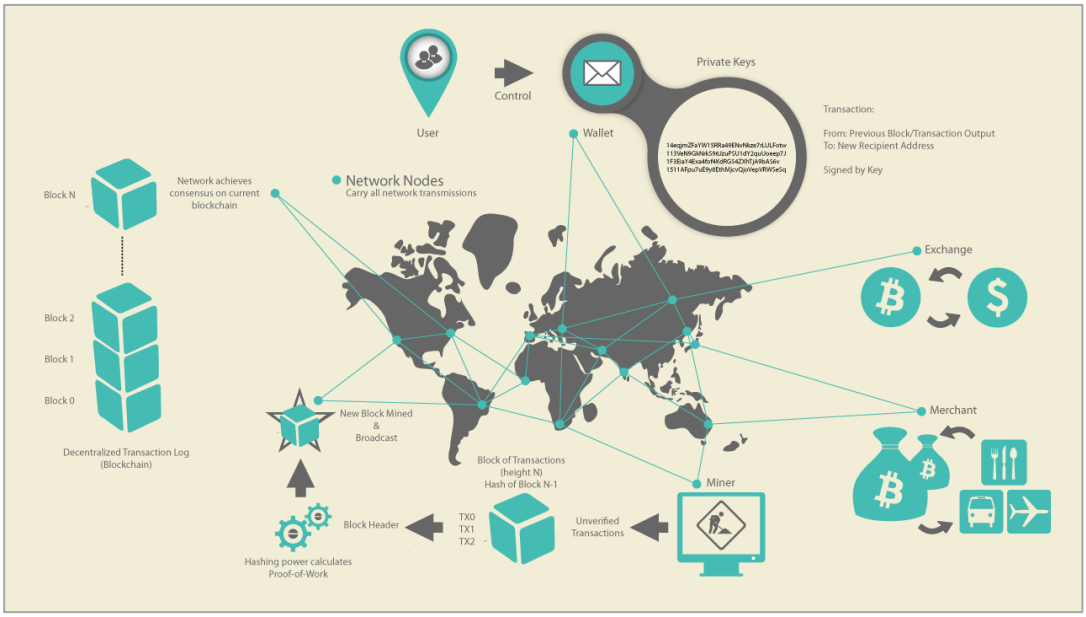

Figure 1. Bitcoin Network Overview (Antonopoulos, Mastering Bitcoin 16).

2.4. Users Addresses and Keys

Cryptography is required to protect the bitcoin network since is it an open network and anyone can explore all transactions recorded into the blockchain. Each user owns a wallet that generate a *bitcoin address* used similar to banks IBAN account address. It is safe to share the bitcoin address since it was generated privately using *digital keys*. For the user to be able to start a payment transaction, she is required to provide a *digital signature*. If a hacker managed to get a copy of the private keys, the hacker can withdraw the funds and they are lost forever.

2.4.1. Public Key Cryptography

There are two types of cryptography systems used nowadays. Symmetric cryptography uses the same secret key to perform encryption and decryption while the public key cryptography (also known as asymmetric cryptography) uses different keys for encryption and decryption (Martin 22). The idea of public-key cryptography is based on using public key for encryption and secret private key for decryption that only the owner knows. The most popular public-key encryption is the RSA (Martin 160-163).

Bitcoin uses Elliptic Curve public key cryptography which is faster, more secure and produces shorter keys when compared to RSA. Usually elliptic curve is defined over a finite field of prime number or power of primes using the cubic function

$$y^3 = x^2 + ax + b \qquad \text{(Eq. 1)}$$

Beside x and y, where (x,y) represents a point on the curve, another single point called "Point at Infinity" make up the elliptic curve. Points on the curve have set of arithmetic rules. Let's assume two points P, Q ∈ E (Elliptic curve). If *P* is the point at infinity, then

$$P + Q = Q \qquad where - P = 0 \qquad \text{(Eq. 2)}$$

Assuming $P \neq 0$ then -P is a mirror point on the curve where the P coordinate is (x,y) and -P coordinate is (x,-y). If P and Q have different x value, then adding them results in another point R that intersects the curve. If a line drawn over the three points is tangent to the curve at point P, then $R = P$. The same rule applies to the point Q (Vagle).

In case of $Q = -P$ then $P + Q = Point\ at\ Infinity$ and in the case of P = Q then the tangent line is over P where $P + Q = R$ (Vagle)

The first step before applying elliptic curve cryptography is to generate private key. Private key is the most important key for the users. It must be stored securely because it is associated with the bitcoin that the user own. Private key is also used to generate the digital signature required for payments. Private key is generated randomly and must between 1 and n-1 where $n = 1.158 * 10^{77}$. The random generated number has the size of 256-bit and represented as a string of hexadecimal format using SHA-256 hash algorithm (1E99423A4ED27608A15A2616A2B0E9E52CED330AC530EDCC32C8FFC6A526AEDD) (Antonopoulos, Mastering Bitcoin 56-59).

Bitcoin uses specific elliptic curve known as the secp256k1 which is defined using the following formula over a finite field of large prime order ($1.158 * 10^{77}$):

$$y^2 = x^3 + 7\ mod\ p \quad \text{(Eq. 3)}$$

The public key generation formula is defined as:

$$Q = k * G \qquad \text{(Eq. 4)}$$

Where Q is the public key, k is the private key and G is a predefined fixed point at the elliptic curve called the *generation point*. When the private key (k) is multiplied by the generation point (G), the public key Q will result to another point (x,y) at the elliptic curve (Antonopoulos, Mastering Bitcoin 62-64).

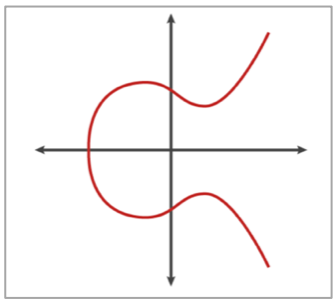

Figure 2. An Elliptic curve (Antonopoulos, Mastering Bitcoin 60).

2.4.2. Digital Signature

Digital signature enables the receiver to authenticate the message and prevents the sender from denying the contents. Digital signature also used to validate the message integrity against data corruption or malicious attack. Figure 3 illustrates how digital signatures are used in public key cryptography (Anton Badev).

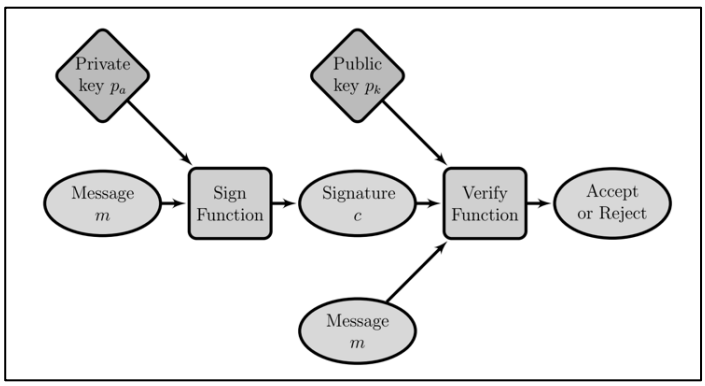

Figure 3. Digital Signature in Public Key Cryptography (Anton Badev).

Bitcoin network uses *Elliptic Curve Digital Signature Algorithm* (ECDSA). ECDSA signature is made of pair of values *S* and *R*. To calculate *R*, random number *k* is selected and then Equation 4 is used to generate temporary ephemeral public key *Qe* which is composed of pair of coordinate (x,y). The value of *x* is equal to *R*. The value of *S* can be calculated using the following equation where *z* is hash of the message, *dA* is the private key, and p is the prime order of the elliptic curve:

$$S = k^{-1}(z + dA * R)\ mod\ p \qquad \text{(Eq. 5)}$$

The verifying process at the receiver side can be calculated using the following equation where Q is the sender public key:

$$Qe = S^{-1} * z * G + S^{-1} * R * Q \qquad \text{(Eq. 6)}$$

If the value of x of the ephemeral public key *Qe* equals to *R*, then the signature is valid (instructables.com).

2.4.3. Bitcoin Addresses

Bitcoin address serves the same purpose of the beneficiary account number in traditional bank transfer between two persons. For example, If Alice wants to receive 0.001 BTC from Bob, then she must first share her bitcoin address with Bob who uses it as the receiver address in the transaction. Bitcoin addresses can be generated from private keys or more popularly from public keys. In newer versions of the bitcoin client software (*Bitcoin Core*) it is also possible to generate the address from a script known as Pay-to-Script-Hash (P2SH) (Antonopoulos, Mastering Bitcoin 64-65).

To generate the address, SHA-256, one-way hashing algorithm, is applied to either the public key, private key or the script. SHA-256 generates 256-bit size fingerprint that is hashed against RIPEMD-160 (another one-way hashing algorithm) which generates 160-bit size fingerprint.

$$RIPEMD160(SHA256(m)) \quad \text{(Eq.7)}$$

The *data* output of this phase is prefixed with a version number (a single byte) which helps in identifying whether the private key, public key or the script was the used to generate the address. For example, 0x00h is used for public key, 0x08h for private key

and 0x05h for P2HS. There are other version numbers in the current release of Bitcoin Core such as 0x0142h that represent encrypted private key.

The next step is to generate a 4-byte checksum by applying SHA-256 twice to a *copy* of the data + prefix generated from the previous step. Only the first 4 bytes are extracted and added to *original* data + prefix. The checksum is used for data integrity where the Bitcoin Core client can validate if the address is correct or not. To make the output *prefix* + *data* + *checksum* easier to read and write for users, the output is encoded using Base-58. Base-58 encoding developed to transfer binary format to text-based output using English alphabet and numbers between one and nine. Base-58 encoding helps users to share their bitcoin address easier since it may result in funds lost. Base-58 eliminates alphabets or numbers that usually appear similar to each other. Capital O might look similar to zero 0 and therefore both are discarded. Also, lower l and capital I might confuse users and therefore are discarded as well (Antonopoulos, Mastering Bitcoin 64-70). The following address is a valid bitcoin address where the prefix 0x00h was encoded to number 1 to indicate this address was generated from a public key (1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy).

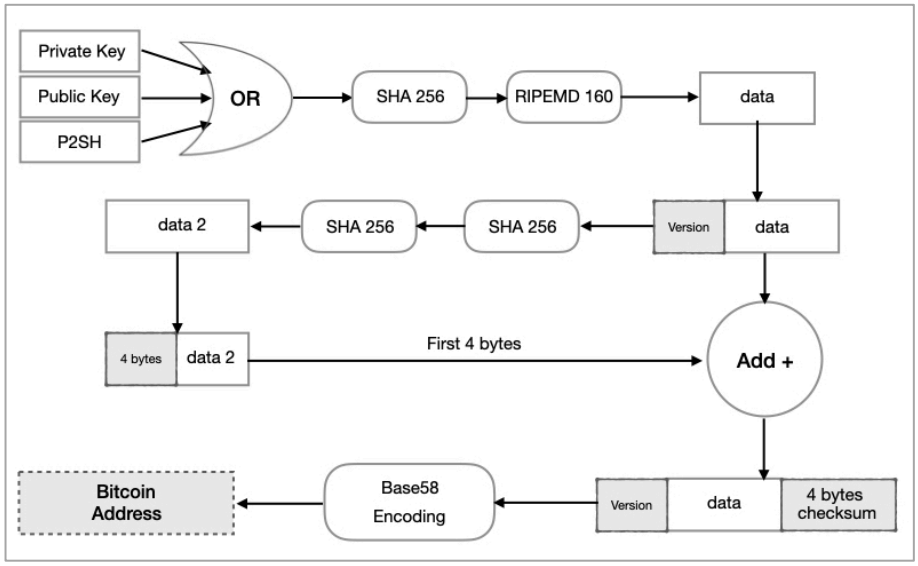

Figure 4. Generating Bitcoin Address.

P2SH address offers flexibility in establishing payments where the beneficiary is not a single user. It is generated from a script file (a simple code or commands) that specifies the rules of the receiver's address. For example, in the case when it is possible for one or two users to receive the funds, the script would include two public keys where any key can be used to validate and receive the payment. This is similar to a joint bank account run by a user and his spouse. This type of script is referred as 1-of-2 signatures. It is also possible to create 2-of-3 signatures script when it is required for two users in a company to use their public keys to receive the funds (Antonopoulos, Mastering Bitcoin 81-82).

2.5. Bitcoin Wallets

Bitcoin wallet is used to manage user's keys and addresses and it does not store bitcoin. Bitcoin wallet must be kept secure and protected from thieves and hackers otherwise the owner might lose her funds forever. There are few types of bitcoin wallets such as paper wallet, software wallet or hardware wallet. Paper wallets are popular because they are stored offline and therefore considered secure against the hackers. Unlike software wallets, paper wallets (printed) keys must be imported into third party application or website for users to be able to check the balance or spend the funds and this might create risk if the user import keys into a compromised third-party software or website (Acheson).

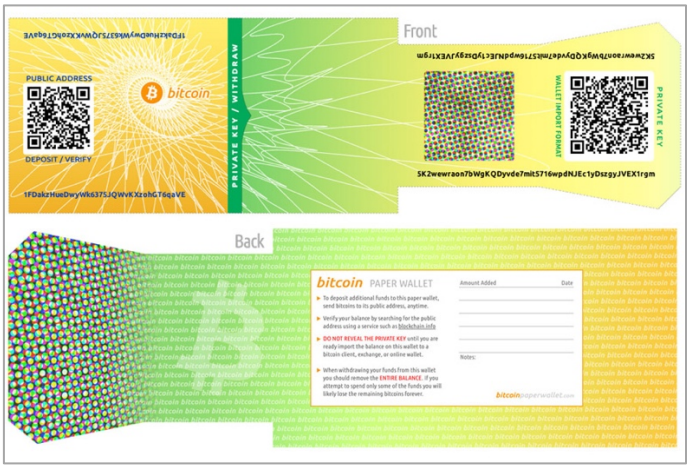

Figure 5. Bitcoin Paper Waller (bitcoinpaperwallet).

Software wallet is the most flexible and easier to use but also can be the least secure since it is connected online and therefore like any other software, might be hacked. Software wallet can calculate user balance from the bitcoin network. Software wallet comes in variety of desktop application, mobile application or websites. Users of software wallets must educate themselves on security before they start using them by following the instructions on the official *bitcoin.org* website (Securing your wallet).

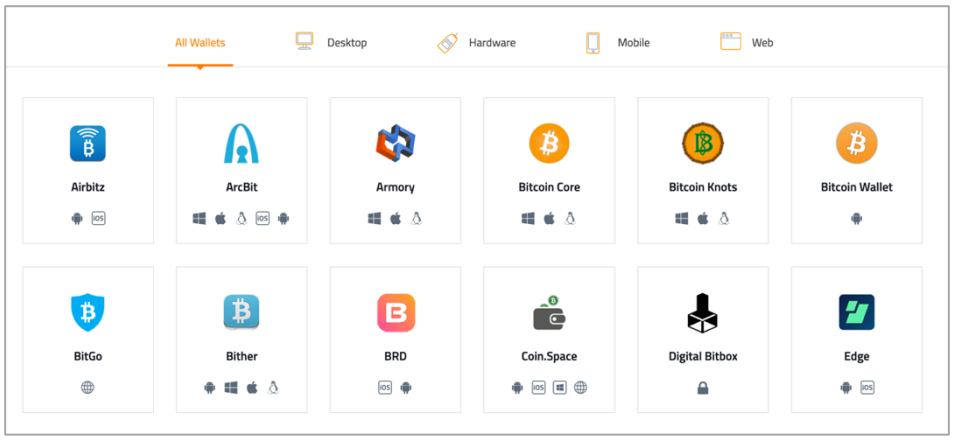

Figure 6. List of Popular Wallets (Bitcoin.org).

Hardware wallets are considered the most secure. They store keys offline and offer software or website for managing payments. Hardware wallets are usually USB devices with range of security measures against malware and loss of keys. One measure is to make private key not readable by the software and inaccessible by the computer operating system (BitcoinWiki, Hardware wallet).

Figure 7. Bitcoin Hardware Wallet (BitcoinWiki, Hardware wallet).

2.5.1 Deterministic vs Nondeterministic Wallets

Private key is used to generate public key that can be used to perform payments. It is possible for a user who owns 100 BTC to have 100 private keys where each key is associated with one bitcoin. In this case, the private key and the associated public key can be only used to spend that one bitcoin or part of it. This creates level of protection to the funds so that if one private key was lost, the user would still own 99 BTC. Private keys are the most important data that must be generated in secure and random methods. Electronic wallets (software or hardware) generate private keys randomly using two different methods. The first and older method is nondeterministic in which the wallets generate private keys randomly without any relations to each other. This type of wallet, despite being secure, is not recommended since it is very hard to maintain, recover, or backup. On the other side, deterministic wallets generate private keys randomly but at the same time they all are driven from a parent random number key known as *seed*. This method opens the possibility of recovering private keys if the seed is known (Antonopoulos, Mastering Bitcoin 93-95).

HD wallet is one form of deterministic wallets that uses tree structure to generate private keys as illustrated in Figure 8. Another feature of HD-Wallets is the possibility to recover or export the same private and public keys without copying them by regenerating the seed using *mnemonic codes*. Mnemonic codes are sequence of English words that the wallet generates randomly at first use and the owner must write them down on a paper and store them on a secure place. Hardware wallets usually use an ordered list 12 or 24 words selected randomly from the dictionary. When the owner wants to export the keys to another HD-Wallet or to recover the lost keys then all she has to do is to enter the same list of words in the same order to regenerate the seed and the associated private keys (Antonopoulos, Mastering Bitcoin 96-97).

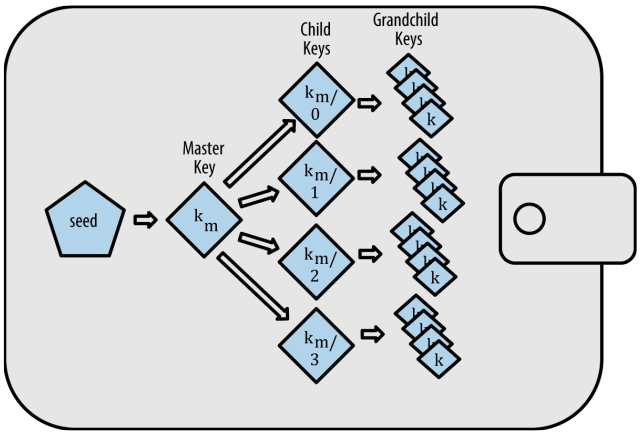

Figure 8. HD Deterministic Wallet (Antonopoulos, Mastering Bitcoin 96).

One method to generate the seed uses a function that takes two parameters and performs 2048 rounds of hashing to produce 512-bit seed. The first parameter is an entropy of a size up to 256 bits generated from the mnemonic words. The second parameter is a password entered by the owner. The password is concatenated with the word "mnemonic" for extra protection against brute force attack. The output seed is then hashed again using HMAC-SHA512 algorithm which produce output of 512-bit size. The left 256-bit is used to generate the *master private key* and the right 256-bit is used to generate the *master chain code*. The purpose of the chain code is to introduce deterministic random data. The master public key is driven from the master private key using elliptic curve cryptography. These three keys plus an index number (0,1,2, 3, …) are fed into HMAC-SHA512 again to generate child private keys and child chain code. By using other index numbers, other nondeterministic child keys are generated. The child keys are used to generate public keys and bitcoin addresses. This technique protects both the seed and the master private key (Antonopoulos, Mastering Bitcoin 99-108).

2.6. Bitcoin Transactions

Transactions are the most important part of the bitcoin network. Transactions represent the records of transferring bitcoin between users in forms of payments, donations or exchange with fiat currencies and therefore, they must be handled well and validated before being recorded to the blockchain ledger.

To be able understand how transactions work, let's assume the scenario illustrated in Figure 9. Alice wants to buy a car from Bob with the price of 4.5 BTC. Alice uses her mobile wallet to create a new payment. Alice's wallet scans the blockchain to calculate her balance by searching for all previous transactions stored in the blockchain owned by Alice's private keys. Alice balance is aggregated from all spendable transactions known as *unspent transaction outputs* (*UTXO*). Next, the wallet constructs a new transaction using three UTXO as inputs accumulating the value of 5 BTC. The wallet uses three UTXO to construct the transaction since UTXO values are indivisible and Alice does not own a UTXO with exactly the value 4.5 BTC or higher amount. The transaction creates two outputs from the inputs. The first output is a transaction with the value of 4.5 BTC assigned to Bob's address and the second output is the *change* back to Alice address. This concept is similar when a customer buys a cup of coffee from the cafe that cost €1 and pays using €5 bill. The shop would then return €4 back to the customer as the change. However, there is extra cost in the case of bitcoin and that is the miner fees. If the second output TX5 of 0.49 BTC was not created, then the change 0.5 BTC will be completely consumed by the miner as mining fees. The value 0.5 BTC is considered very expensive for the fees and therefore, only 0.01 BTC are set for fees from accumulating the outputs (5.0 – (4.5 + 0.49) = 0.01 BTC). After inserting the transaction into the blockchain, Bob's wallet detects new UTXO assigned to his address and therefore is able to confirm Alice's payments and update his balance. Now, Bob wants to buy a bicycle with the price of 0.1 BTC. Bob's mobile wallet constructs a new transaction using the output from Alice's transaction TX4 as the input for the new transaction. Only one input is required since the value of the input is higher than the cost of the bicycle. The same process is repeated until the new transaction is stored in the blockchain (Antonopoulos, Mastering Bitcoin 117-120).

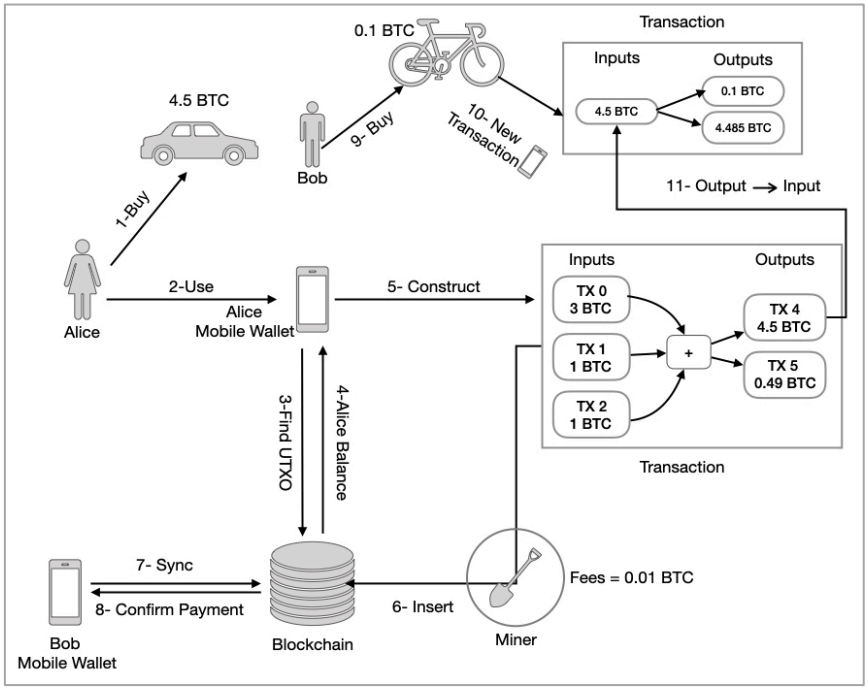

Figure 9. Transactions Overview.

From technical perspective, transaction is a data structure composed of few fields as illustrated in Figure 10. For simplicity, only a single input and output are used to explain the transaction. The first field is the version number that identifies the type of the data structure. This is essential for the transaction validation process by nodes and miners. The second field is the number of inputs. For example, in the case of Alice purchasing Bob's car, the number would be equal to three. Next, the input transaction field which is composed of few other fields. The first two fields are the transaction Id and the index of the previous output transaction (output becomes input). Input field contains a signature script called *ScriptSig* that proofs the ownership of the output and another field that specifies the script size. The last field is a sequence number that was added initially to indicate that the transaction is not finalized and will be replaced with another transaction of higher sequence number. However, this feature was not implemented in the Bitcoin Core and the value is normally set to 2^{32}. In current versions of Bitcoin Core, the sequence number field can be used as a setting input for other fields like the *LockTime* field. When it is set to value less than 2^{32}, the time lock will be treated as a relative time otherwise it is an absolute time in future. LockTime field prevents the receiver from spending the output before due date (Valentin Vallois).

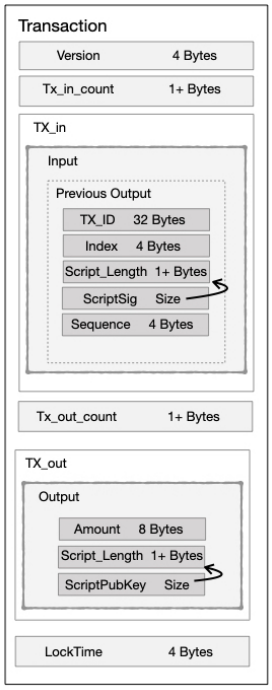

Figure 10. Transaction Data Structure (Valentin Vallois).

Output counter field comes next followed by output transaction field. Output transaction field contains few more fields. First, it contains the value of spendable bitcoin in *satoshis*. Since fiat currencies are made of units and subunits (For example, dollar and cents), bitcoin is also made of subunit called satoshi. For example, the value of 1,500,000 satoshi equals to 0.015 BTC. As explained earlier, output's ownership must be set through by specifying the receiver public key therefore the output field contains a field that identifies the size of the following field and a public key script *ScriptPubKey* (Valentin Vallois).

Satoshi Nakamoto designed the bitcoin script programming to be very simple and restrictive to protect the bitcoin networks from malicious attacks. Originally, ScriptSig contained only the digital signature and the ScriptPubKey contained the public key or bitcoin address of the receiver. Later, developers of the Bitcoin Core software made it possible to use both scripts to perform advanced payment like the multi-signature options in P2SH address discussed previously (Valentin Vallois).

2.6.1. Transaction Scripts

On the current version of Bitcoin Core, the ScriptPubKey is known as a *locking script* because it contains a condition or a quiz that must be solved before the output become spendable. To solve the quiz and unlock the script, ScriptSig is used and therefore it is known as the *unlocking script*. Scripts are executed in stack based Last-In-First-Out (LIFO) queue. The simplest type of locking scripts is known as *Pay-To-Public-Key-Hash* (P2PKH) where the quiz is solved by matching the public key of the receiver with the digital signature. The following example uses P2PKH script:

Listing 2.1

```
OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY OP_CHECKSIG
```

Then the unlocking script <Owner Signature> <Owner Public Key> is combined with the locking script as:

Listing 2.2

```
<Owner Signature> <Owner Public Key> OP_DUP OP_HASH160 <pubKeyHash> OP_EQUALVERIFY
OP_CHECKSIG
```

The script is executed in the following order. The owner's signature will be pushed into the stack followed by the owner public key. Next, the public key from previous step is duplicated using OP_DUP command. The fourth step is to hash the duplicated copy using OP_HASH160 command that applies Equation 7. In the fifth step, the supplied value of <pubKeyHash> is pushed to the top of the stack and then is compared against the results of step 4 using OP_EQUALVERIFY command. If the validation succeeds, the results of step 4 and 5 are popped out of the stack. In the final step, OP_CHECKSIG command compare the values of step 1 and 2 (using Eq. 6). If the validation succeeds, both values are popped out of the stack and TRUE is pushed to the stack that represents the final result of the execution process (Antonopoulos, Mastering Bitcoin 132-138).

There are more commands available for complex scripts. For example, in the case of multi-signature script where two of three (2-of-3 script) public keys are required to unlock the locking script, then the script would look similar to:

Listing 2.3

```
<Signature B> <Signature C> OP_2 <Public Key A> <Public Key B> <Public Key C> OP_3 CHECKMULTISIG
```

However, the script can become cumbersome and complex to execute if the number of public keys that are allowed to unlock the script is large. Another issue, the miner fees can be determined by the script size since larger scripts need more computing power and RAM to execute. To solve this issue, P2SH was introduced to reduce the complexity and the size of the *locking script* by hashing it (Eq. 7) and encoding it using Base-58 and then the output's size equals to 20-byte only. Listing 2.4 is called the *redeem script* and Listing 2.5 is the locking script.

Listing 2.4

```
OP_2 <Public Key A> <Public Key B> <Public Key C> OP_3 CHECKMULTISIG
```

Listing 2.5

```
HASH160 <20-byte hash of redeem script> EQUAL
```

The locking script would still contain the signatures of B and C as in Listing 2.3. When Bobs wants to spend the funds, the following scripts are executed on his machine in order:

Listing 2.6

```
<Redeem Script> HASH160 <20-byte hash of redeem script> EQUAL
```

Listing 2.7

```
<Signature B> <Signature C> <Redeem Script>
```

Listing 2.6 first check if the hashing of the redeem script is equal to the hash stored in the locking script. Next, unlocking the script is executed using pair of digital signatures and three public keys similar to Listing 2.3. The key difference is that the redeem script is not stored in the transaction itself but calculated at the receiver machine (Antonopoulos, Mastering Bitcoin 149-153).

2.7. The Blockchain

The blockchain is a temper-proof ledger made of *blocks* (data structure) linked together as a *chain*. The chain protects the integrity of the transactions in each block. Each block is identified using a fingerprint (hash) that was generated by feeding a hashing algorithm with the data contents of the block. Every block stores two fingerprints; its own hash and the previous block hash and that is how the chain is formed as illustrated in Figure 11. It is not possible to alter block contents since the hashing algorithm would generate different hash that invalidates the original hash stored in the previous block and as a result the whole ledger become invalid (Orcutt). The only exception where a block stores only its own hash the first block and is called the *genesis block* which was created in 2009 by Satoshi Nakamoto (Antonopoulos 2017, 199).

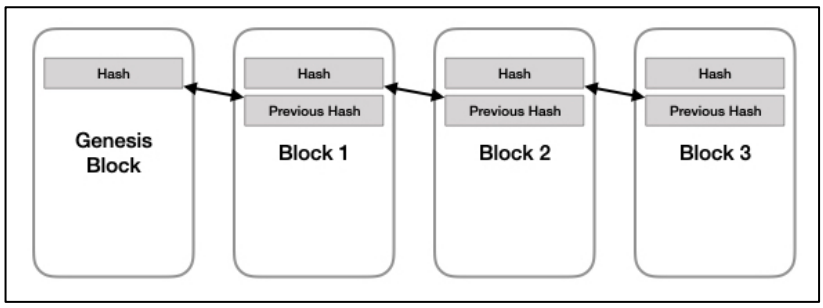

Figure 11. The Blockchain Ledger.

The first field in the block identifies the block size. The second field is the *block header* that contains metadata entries such as version number, previous block fingerprint and the *merkle root* hash. There are also *timestamp*, *difficulty target* and *nonce* which are used in mining process and will be discussed in the next section. *Transaction count* field is used to identify how many transactions the block stores. Single block can store more than 500 transactions where each transaction has the size of at least 250 bytes. The last field contains the transactions list. The *coinbase* transaction is the first transaction stored in the transactions list and it is created by the miners as the reward for their mining effort. In the genesis block, the coinbase transaction contains a hidden message: "The Times 03/Jan/2009 Chancellor on brink of second bail-out for banks." The message was referencing a headline from the British newspaper The Times during the period of financial crisis due to bank debuts that resulted from housing mortgages (Antonopoulos, Mastering Bitcoin 196-199).

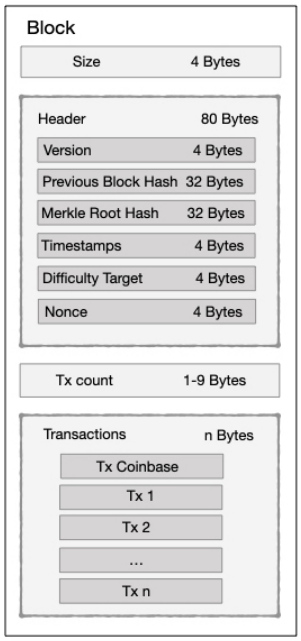

Figure 12. Block Data Structure (Antonopoulos, Mastering Bitcoin 196-197).

We mentioned before that each block contains two fingerprints that are used identify the block itself and the previous block. But in reality, the block's header contains only

previous block hash field. The block hash is not stored in the block, but it is calculated using the following equation:

$$SHA256(SHA256(m)) \qquad \text{(Eq.8)}$$

The block hash from the previous equation is called sometimes the *block header hash* because the header contents were used to generate it (Antonopoulos, Mastering Bitcoin 197-198).

2.7.1. Merkle Trees

Merkle trees are cryptographic data structure where each node in the tree has two child nodes known as *left* and *right* nodes. Binary trees are composed of *leaf* nodes and a single *root* node. Similar to binary trees, the purpose of the merkle tree is to optimize the searching algorithm used to find if a single transaction is stored in the block or not. To create the merkle tree, Equation 8 is applied to each node. Each pair (left and right nodes) hashes are combined together and then Equation 8 is applied again until the root is created from combining the last two branches as illustrated in Figure 13. (Antonopoulos, Mastering Bitcoin 201-202)

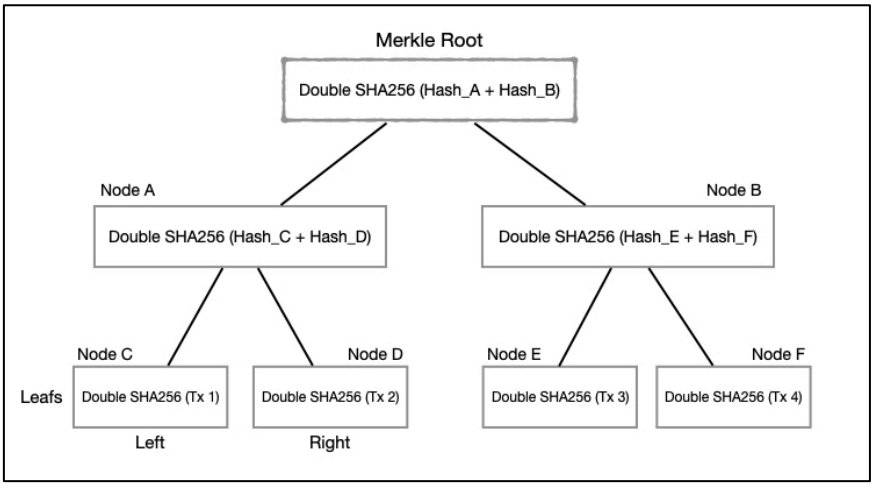

Figure 13. Merkle Tree.

By storing the merkle root in the block header, it becomes possible for mobile wallets to download only blocks' headers to validate if a certain transaction is stored in the block and if it is valid. Mobile wallets are known as SPV nodes because they only download blocks' headers instead of the whole large-sized blockchain. Nodes that store the whole blockchain are known as *full nodes*. Full nodes supply SPV nodes with block header and a *merkle path* to assist SPV in verifying the transaction. For example, if SPV node has transaction (TX_K) and wants to verify that if it is a valid transaction, then full node sends the block header along a merkle path that consists of four hashes (H_L, H_{IJ}, H_{MNOP}, $H_{ABCDEFGH}$) highlighted in blue color in Figure 14. The SPV node performs the following steps to validate Tx_k:

1- Apply Eq.8 on Tx_k that produces hash H_k
2- Apply Eq.8 on H_k and H_L that produces H_{KL}
3- Apply Eq.8 on H_{kL} and H_{IJ} that produces H_{IJKL}
4- Apply Eq.8 on H_{IJKL} and H_{MNOP} that produces $H_{IJKLMNOP}$
5- Apply Eq.8 on $H_{IJKLMNOP}$ and $H_{MABCDEFGHNOP}$ that produces merkle root

If the merkle root that was calculated in step 5 is equal to the merkle root stored in the block header, then the transaction is valid (Antonopoulos, Mastering Bitcoin 203-207).

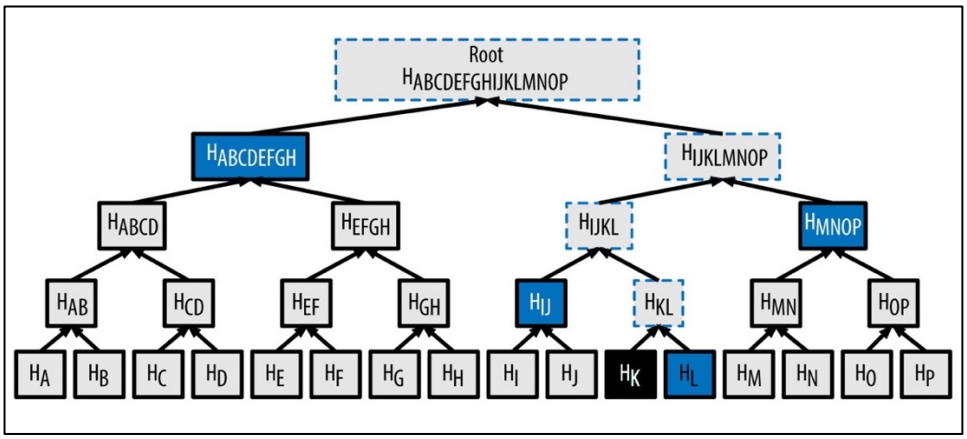

Figure 14. Merkle Path (Antonopoulos, Mastering Bitcoin 204).

2.8. Bitcoin Mining

Mining is the smartest innovation of the bitcoin system. Mining secures the bitcoin system by establishing *trust* among the nodes participating in the network through consensus protocol. Mining is very important since the bitcoin network running without a central authority to validate the transactions and to reward the miners. Mining is expensive operation in the terms of processing power and usually takes about 10 minutes of work from the miner to insert a new block into the blockchain. Miners compete against each other in solving a cryptographic puzzle to win the reward and earn transactions fees. Only one miner can win the challenge and then a new competition starts for the next block. The solution to the cryptographic puzzle algorithm is stored in the block as a proof that the miner has performed expensive process which is known as *Proof of Work* (PoW). Mining is the process that generate new bitcoin supply similar to how the central bank issue new money via printing money notes. Miner rewards for finding the PoW solution decrease every 210,000 blocks or almost every four years. It started as 50 BTC in 2009 and nowadays the reward is 12.5 BTC. By 2140, bitcoin supply will come to an end and miners may then earn only the transactions fees. This limited fixed-rate supply makes bitcoin a deflationary currency. There have been strong debates in financial sectors over the advantages and disadvantages of both the deflationary and inflationary currencies (Antonopoulos, Mastering Bitcoin 213-216). The energy consumed by the miners to find PoW solutions every year is almost similar to the energy Finland consumes per year.

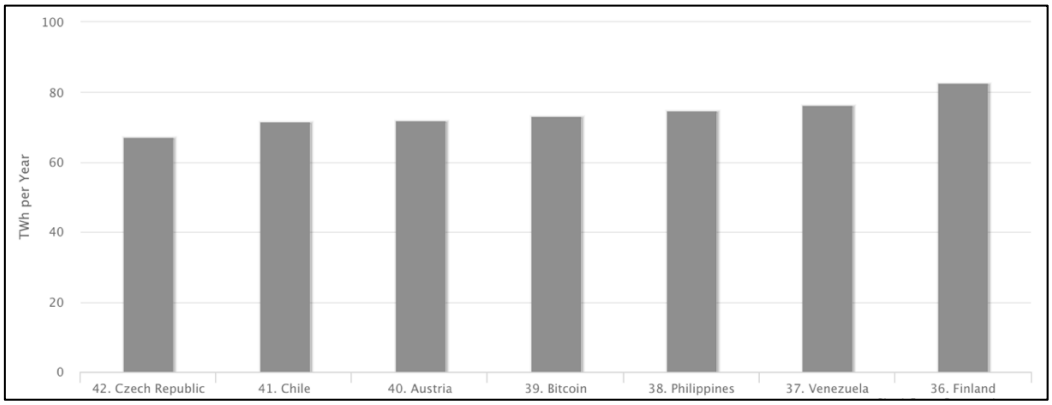

Figure 15. Bitcoin Energy Consumption Chart (Digiconomist).

2.8.1. Decentralized Consensus Protocol

Bitcoin consensus protocol establishes trust between thousands of nodes in the network using four stages. First, all nodes validate every transaction propagated into to the network against comprehensive list of rules. When transaction is validated, each node stores the transaction in a temporary place known as *memory pool* or *mempool*. Next, miners select transactions with higher fees from the mempool to build a *candidate block*. Miner then adds the mining reward plus the accumulated transactions fees into the coinbase transaction. Unlike regular transactions, coinbase has no inputs and only a single output paid to the miner address. In the third stage, mining competition starts between miners to solve the PoW puzzle. The first miner to solve the puzzle will transmit the new block to all peer nodes where it is validated against set of rules and then propagated across the network. After the validation of the candidate block, each node inserts the new block to the top of the blockchain. The final stage of the consensus protocol involves finding a solution to the situation where two miners solve the PoW puzzle at the same time and propagate their blocks across the network. This situation is known as *blockchain forks* and it will be discussed shortly (Antonopoulos, Mastering Bitcoin 217-240).

2.8.2. Proof-of-Work Challenge

PoW challenge is a guessing game where the miners may only find the solution by using brute force technique. Single miner might need to guess quadrillions of times before successfully it can find the solution. We mentioned before that the block fingerprint is not stored in the header, but rather it is calculated from the header contents. We also mentioned that the header contains three fields used in mining process *timestamp, difficulty target and nonce*. Timestamp is set by the miner for the time the mining process start. It is used later by other nodes to validate the block time when the miner claims the right solution. The challenge is set by using a *difficulty target* and can only be solved by guessing the *nonce*. The difficulty target is adjusted by the all the

nodes in the network every 2016 blocks to keep mining process at fixed rate of 10 minutes per block (Antonopoulos, Mastering Bitcoin 228-236).

The challenge for guessing the right fingerprint for the header can be explained better using simpler contents such as "Hello, world!" text. Let's assume that the difficulty target is a 32-bits hexadecimal number that must start with at least three zeros. The nonce is added to the end of the text and then both are hashed using SHA256 algorithm and the result (represented in hexadecimal format) is compared against the difficulty target. If the result starts with at least three zeros, then the PoW solution is found, and it is equal to the *nonce value* otherwise, the nonce is incremented and the process starts again. Table 1 shows that 4251 attempts were necessary to find a fingerprint that satisfies the difficulty target and the nonce value (equals to 4250) is recorded into the blockchain header as the solution (BitcoinWiki, Proof of work). While finding the solution is difficult task, validating it is very simple. Only one single computation is required by the other nodes receiving the winner block.

Table 1. Solving "Hello, World!" PoW (BitcoinWiki, Proof of work).

Input	Nonce	Hex	Valid
Hello, world!0	0	1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934	No
Hello, world!1	1	e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a	No
Hello, world!2	2	ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd426	No
…	..	…	No
Hello, world!4250	4250	0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4	**YES**

In reality, the PoW challenge is much harder than the example above. Miners use hundreds or thousands of specialized *Application-Specific Integrated Circuit* (ASIC) connected together to find the solution faster than competitors. As a result, this has increased the difficulty target and made it impossible for individual miners to compete unless they join a mining pool where participants cooperate together to the find the solution (Hoffman).

2.8.3. Blockchain Forks

Blockchain forks is an event when the blockchain is split into branches due to several circumstances. The most common and simplest fork occur when a node in the bitcoin network receives a new block that cannot be placed to the top of the blockchain because another block with similar *previous block hash* arrived earlier. This is the result of two miners finding the solution to the PoW puzzle at the same time and propagate their blocks to the network. The node does not discard the block but instead creates a new branch until the bitcoin consensus protocol resolve this issue. There are other types of forks known as *soft forks* and *hard forks* (Castor).

The simplest fork is temporary situation that is usually resolved after the next block is mined. Let's assume bitcoin network has five nodes. Each node contains a copy of the blockchain and performs mining as well. At the beginning, all nodes are in synchronization where the last block (*block-K*) stored in the blockchain is at the top on every blockchain. Next, both *node-B* and *node-A* solve the PoW puzzle at the same time and propagate two different blocks to the network (*block-J* and *block-R*). Now, *node-C* receives *block-J* faster from *node-B* while *node-E* receives *block-R* faster from *node-A* and then *node-E* propagates *block-R* to *node-D*. As a result, the blockchain is split into two copies with different header block stored on (*node-A*, *node-E* and *node-D*) and another different header block stored on (*node-B* and *node-C*). Shortly, node-C receives *block-R* from *node-A* which has similar *previous block hash* to the *block-J*. At this moment, *node-C* become aware that a fork has occurred and creates a new branch for *block-R*. All other nodes receive the alternative block and creates branches as well as illustrated in Figure 16 (Antonopoulos, Mastering Bitcoin 240-247).

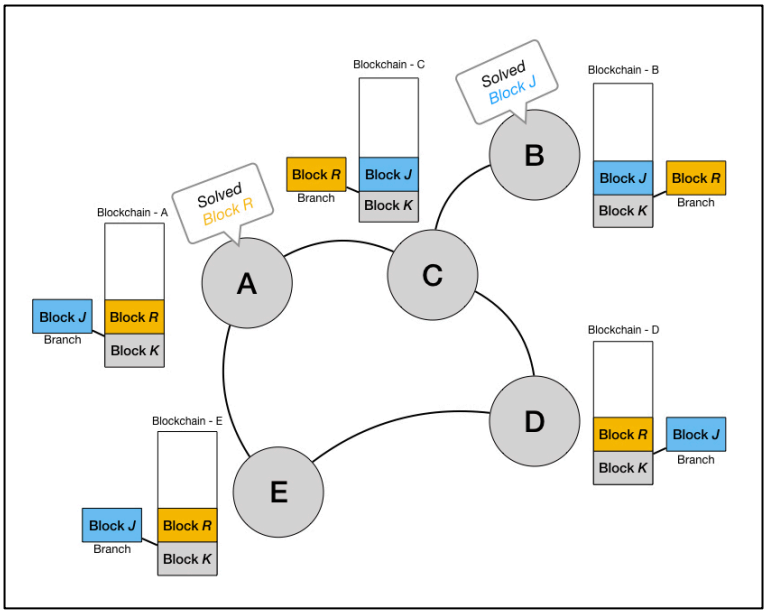

Figure 16. Temporary Blockchain Fork.

The fork can be resolved after the next mining challenge ends. In this scenario, *node-D* finds the next PoW before any other node in the network and propagates *block-N* to all nodes. *Block-N* contains a *previous block hash* to *block-R*. All nodes in the network accept that the *block-R* is the winner from the previous challenge and discard *block-J*. *Node-B* has lost its winner block and therefore lost the reward and transactions fees. Any transactions that were recorded into *block-J* and were not recorded into either *block-R* or *block-N* are returned to mempool. Finally, all blockchain copies are in synchronization as illustrated in Figure 17 (Antonopoulos, Mastering Bitcoin 240-247).

Blockchain forks may case *double spending* problem where attacker uses the input transactions more than once. Usually, user selling expensive items must wait until she receives confirmation that the transaction was mined into the blockchain. But if the transaction was mined into a fork branch that gets discarded later, then the transaction is sent back to the mempool. In this case, the buyer may take an advantage and use the same transaction with higher fees to purchase another item. The miner will pick the new transaction which invalidate the previous one. Therefore, it is recommended to wait for at least six blocks to be mined on the top of the block that contains the user transaction (Antonopoulos, Mastering Bitcoin 253-256).

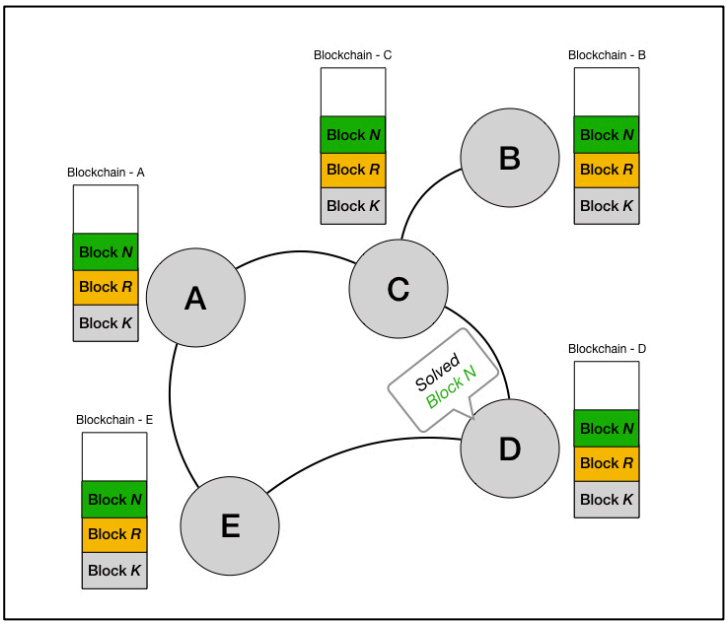

Figure 17. Temporary Blockchain Fork Resolved.

Hard fork means that the branches of blockchain may not be resolved automatically. For example, if some nodes decide to adopt an alternative PoW challenge that consumes much less energy while the rest nodes reject the proposal then the blockchain is split into two branches that both keep growing in parallel. Hard forks are considered risky since they may only be resolved when all nodes agree to new proposal and that is difficult to achieve in a decentralized network with thousands of nodes distributed globally. On the other side, soft fork may happen when some nodes decide to update the list of the validation rules but without disturbing the network. In this scenario, new nodes may reject blocks that are considered valid for older nodes while older nodes would still accept all blocks. Soft forks happened few times in the bitcoin network. For example, when the P2SH was added. Soft forks can be resolved when the majority of the nodes update to the newer system which forces others to update as well. Otherwise, older nodes would always loose the fork and therefore loose own mined blocks (CoinDesk).

2.9. Consensus Attack

Blockchain forks may be used as an attack mechanism against top recent blocks to invalidate them and hence the transactions included into them can be become spendable again. This attack is known the “51% Attack” since it may require the majority of the mining power to occur. Imagine the scenario where the malicious attacker Paul purchase very expensive diamond from Bob. Bob’s wallet receives the confirmation that the transaction was mined and handle the diamond to Paul. Bob was not aware than Paul operates a mining pool that controls 51% of the mining power. Paul creates a fork by mining another block that contain the same transaction UTXO used to purchase the diamond but set the receiver address to Paul’s own address instead of Bob. Because of the 51% mining power, it is very likely that Paul mining pool will mine the next block and therefore the blockchain fork resolve into the new chain. Bob’s transaction is sent back to mempool and later become invalid. To avoid this attack, Bob should have waited for at least six blocks to be mined after he received the confirmation or alternatively wait for 24 hours before handling the diamond to Paul.

Controlling 51% of the mining power makes the probability of the attack very high and almost guaranteed to occur. Controlling smaller percentage means lower probability but the attack is still possible (Antonopoulos, Mastering Bitcoin 253-256). So far, bitcoin has not suffered from similar attack due to the huge mining power used that makes it very expensive to control 51% of the mining power.

2.10. Critiques

Bitcoin concepts are well designed but like any other technology, it suffers drawbacks that may become critical issues in future. The first obvious problem is that who control most of the mining power can control the bitcoin network. This problem creates a super *cryptopower* who can deny specific addresses from using the network by simply ignoring them and leaving the transactions in the mempool. Does this sound familiar?

Isn't it similar to a central bank prohibiting specific users from transferring money for political or personal reasons?

Competing for controlling the mining power means sharp increase in electricity consumptions between different miners or even between countries. For people working on solutions to save the energy, this might be considered as a disaster. Eventually, countries with large sources of energy may dominate the network forcing poor countries to submit to the new financial system or enforce rules and regulations to use the bitcoin. Does not this sound similar to nowadays where super power countries are controlling the current monetary system?

Without regulations in countries, bitcoin growth will be limited. The questions someone might ask, is USA willing to accept replacing the dominance of dollar in the global trade for the sake of bitcoin? Are banks and other financial institutions willing give away their strong influences for the sake of public P2P monetary system? It could be better for people who want to see bitcoin become widely adopted should establish organizations that aim towards achieving local regulations by courts and other law enforcement systems. This enables transparency and trust for more people to join the world of bitcoin.

We discussed earlier the consensus attack and it can be used in double spending problem and that is the seller of expensive diamond should wait for six blocks (one hour) or more to confirm the payment. However, without regulations to protect the buyer, the seller may deny the ownership of the bitcoin address that received the payment and therefore the buyer might lose his bitcoin. If consensus attack starts to become common in bitcoin, serious trust problem will arise between users and then third parties might become involved in the transactions during the wait period. Something the bitcoin wanted to eliminate from the beginning.

Digital currencies have found appreciations by some political leaders who are looking for alternative monetary system where a single tweet from the president of the US could cause the local currency of any country to fall against the dollar. Now the question, will

bitcoin driven by politicians cause a major conflict in the world? It might sound unlikely at the moment, but is it impossible in the future?

Some people believe bitcoin started to become popular for the first time because a darknet website known as Silk Road. The website was the first known case for using bitcoin as method of payment to sell drugs and other illegal materials. The website was shut down by the FBI and the founder received life in prison. Later on, hackers started using bitcoin as a payment for ransomwares. The most popular case was the infamous WannaCry ransomware. Now, another question arises, was the invention of bitcoin by Satoshi Nakamoto as a response for the current financial system that caused financial disasters or as secure payment solution for criminals? Unfortunately, bitcoin may not be able to keep the bad people away but at least it is fair for everyone.

Global cybersecurity problem could occur in future if the current encryption techniques become broken using quantum computing. Bitcoin depends on public key encryption and therefore it might face serious challenges for its existence. Investing in R&D by rich bitcoin owners might help to save the bitcoin in future. Awarding grants in bitcoin for cryptography researchers may result in papers that help in protecting, improving and developing the bitcoin.

Bitcoin could benefit from having a single organization to represent it and defend it against banks and other “enemies”. Establishing such as organization using bitcoin as source of fund is not a difficult task. If this organization succeed, bitcoin value will increase sharply, and bitcoin will find global adoption.

3. ETHEREUM

3.1. Introduction

Ethereum is a platform that extends the concepts of bitcoin to provide the possibility for running decentralized applications known as *DApps*. It is virtually possible to develop DApps to run any kind of transactions or agreements between the users of that particular decentralized application (Ray, Ethereum Introduction). Possibilities include developing a DApp for storing people's ownerships of lands and houses while taking advantage of the secure blockchain ledger and without having a corrupt central authority that may manipulate the ownership records. It is also possible to develop a secure voting DApp for any kind of elections where voters and candidates can trust the system knowing votes are casted correctly without being manipulated by a central authority or hackers. This voting is not only secure and fast, but it would cost fraction of what regular paper ballot elections cost nowadays. Using Ethereum require fees that cannot be paid with fiat money. Ethereum has its own digital currency known as *Ether* (ETH) which is used for payments.

Ethereum comes with big promises and has been referred to with different names. Ethererum is called sometimes the *Web 3.0* because it promises to provide an alternative to the internet where no large companies can control the traffic and the storage of data similar to Amazon or Google Cloud services. Ethereum promises that the internet cannot be censored or blocked by governments or internet provides. Unlike today where the website is usually hosted on single server, future websites will be hosted on thousands or nodes distributed across the globe. Ethereum also referred to as the "World Computer" where all the nodes around the world work together to perform tasks and ensure all the applications are always available and secure (Ray, Ethereum Introduction).

3.2. Ethereum Network Overview

Technically speaking, Ethereum network and its architecture is similar to bitcoin. Similarities include P2P network, full nodes, miners, wallets, PoW and consensus protocol, blockchain ledger and cryptocurrency called Ether. For Ethereum to be able to run applications, new concepts were introduced. There are two types of accounts on the Ethereum network. When a user accesses Ethereum network via wallet for the purposes of using the cryptocurrency similar to bitcoin, then this type of account is called *Externally Owned Account* (EOA). For the purposes of running decentralized applications, there is another type of account called *contract* account. Unlike the EOA account which is controlled by the user via private key, the second type is controlled by the network and is based on the concept of *smart contracts*. Smart contracts are advanced programming scripts (compared to bitcoin scripts) that run on full nodes using special virtual machine called *ethereum virtual machine* (EVM). EVM similar to Java virtual machine, that is able to interpret and run compiled smart contracts (bytecode) regardless or the hosting hardware or operating system. Contract accounts are initialized via EOA accounts by either sending ETH or data to them. Smart contracts can be programmed to send or receive ETH cryptocurrency just like EOA or to perform specifics tasks or even call other smart contracts on the network (Antonopoulos and Wood, Introduction).

Ethereum ability to execute different types of transactions make is vulnerable to DDoS attack. The network could be halted if a hacker deploys a smart contract that performs expensive computations indefinitely. To protect the network, the concept of *gas* was introduced. Gas is consumed by the instructions while executed on EVM. The user of EOA account must specify maximum amount of gas in ETH she is willing to spend for the EVM to execute the transaction. Smart contracts are compiled into bytecode before being executed by the EVM. This bytecode is represented in low level set of instructions similar to assembly language. Each bytecode instruction has a gas cost and therefore the total gas cost of any smart contract can be calculated programmatically. Gas unit cost is not fixed but depends on the network resources. If the network is busy

performing many instructions, the gas cost could rise up exceptionally. The transaction fee is calculated according to:

$$Transaction_{fees} = Gas_{consumed} * Gas_{price} \quad \text{(Eq.9)}$$

If the user set gas amount more to what was needed to complete the execution, then remaining gas will be returned to the user. If the amount was less to what was required, then the gas is consumed by the miner and the instruction is rejected and rolled back by the EVM. Similar to the bitcoin subunit satoshi, gas price is measured in *Wei* where one Ether equals to 10^{18} wei (Infante 59-63).

Ethereum uses similar public key cryptography for generating EOA and contract accounts addresses. The major difference to bitcoin is the use of Keccak-256 hash algorithm instead of SHA-256. Keccak-256 hash algorithm was the winner of the SHA-3 cryptographic hash function computation held by National Institute of Science and Technology (NIST) in 2007. Ethereum implemented the original algorithm of Keccak-256 while NIST decided to modify it before being accepted as the SHA-3 standard. Therefore, the output of the SHA-3 is slightly different from Ethereum Keccak-256 hash implementation. Only the last 20 bytes (LSB in big-endian) output of the Keccak-256 are used to represent an Ethereum address. Ethereum address are stored without checksum and are not encoded with Base58 encoding but rather represented in raw hexadecimal format. The original proposal for Ethereum addresses suggested that checksum and encoding should be implemented at higher layers. However, due to the slow development of Ethereum, some wallets started to implement *Inter exchange Client Address Protocol* (ICAP) encoding. ICAP is compatible with *International Bank Account Number* (IBAN) encoding format. IBAN consist of country code, checksum and bank account number. In Finland, IBAN starts with FI as the country code while *XE* used in the beginning of Ethereum addresses (Antonopoulos and Wood, Keys and Addresses).

3.3. Transactions

Ethereum, as well as bitcoin, can be viewed as *state machine* where each transaction starts with initial state and then passes through incremental executions until it reaches the final state where valid instruction is recorded into the blockchain. The state machine can be represented mathematically according to the following equation where σ is the state of the transaction, T is the transaction and *Y* is the state transition function (Wood).

$$\sigma_{t+1} \equiv Y(\sigma_t, T) \qquad \text{(Eq.10)}$$

When the transaction is a payment between two users, Ethereum will record a *transfer state* of money from the sender to receiver. When the transaction is invoking a smart contract, the EVM will execute the transaction and change its state accordingly. The third type of instruction is used to create new contract account where the transaction contains the bytecode of the smart contract (Antonopoulos and Wood, Transactions).

Regardless of the instruction type, each transaction is composed of seven fields as illustrated in Figure 18. Nonce is a scalar value that represent the number of transactions originated from the EOA address. For the first transaction, nonce is equal to zero and then it is incremented sequentially. Nonce is needed because of the decentralized nature of the network. If an EOA account create two transactions and propagate them to the network but first transaction is more important and must be executed first. The first transaction will contain lower nonce than the second transaction. If a node receives the second transaction before the first one duo to a delay in the network, the node will place it in the mempool and wait for the first transaction to arrive. Nonce is also used to make transaction appear unique when the contents are similar. This uniqueness protects the transaction from being executed multiple times and therefore protect EOA account balance (Antonopoulos and Wood, Transactions).

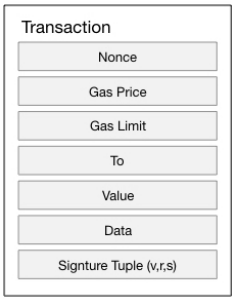

Figure 18. The Structure of Ethereum Transaction.

The gas price and gas limit are used for transaction fees as discussed previously. Gas price changes according to the network traffic. There are popular websites similar to *ethgasstation.info* that provide information about current gas price and other analytic information that can assist the user of the EOA account as illustrated in Figure 19.

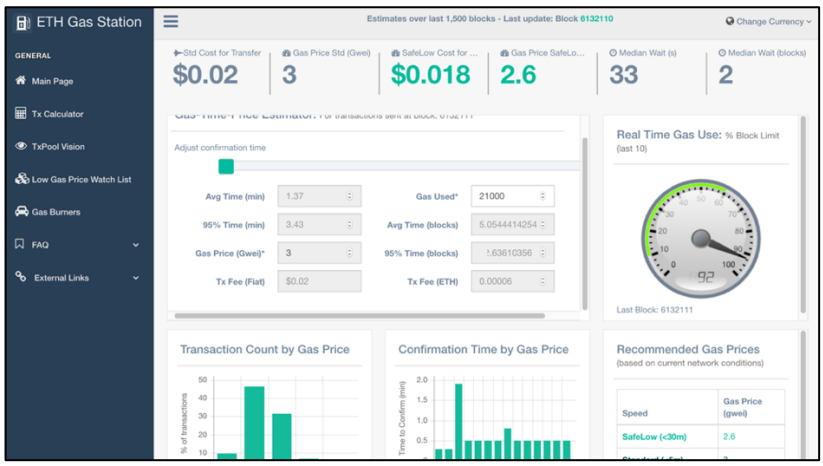

Figure 19. Ethereum Gas Station Website (ethgasstation.info).

The next field (To) is the recipient EOA or contact account address. Validating the address should be done at user level since Ethereum does not offer any validation. Failing to validate the address could result in ETH funds lost. The next two fields determine the type of the transaction. For example, if Bob wants to send ETH to Alice then the transaction *value* field should be set. Otherwise, if Bob wants to call a smart contract, then *data* field is set. The sender address is not stored in the transaction but

rather calculated from the last field. Signature tuple contains three values (v, r, s) where both *R* and *S* are used with digital signatures and they were previously discussed in the section 2.4.2. Since the elliptic curve is symmetric across the x-axis and therefore the sender public key could be one of two possible values, the third value *V* is required. The two possibilities of the public key are calculated according to the following formulas:

$$K_1 = R^{-1}(SW - zG) \qquad \text{(Eq. 11)}$$

$$K_2 = R^{-1}(SW' - zG) \qquad \text{(Eq. 12)}$$

- K_1 and K_2 are the two possible public keys.
- R^{-1} is the multiplicative inverse point of R.
- W and W' are the two possible points of ephemeral public key *Qe* x-axis.
- z is the hashed message.
- G is the elliptic curve generator point.

If the value *V* is even, then the point *W* is the correct point and therefore the sender public key is K_1 otherwise it's the K_2 key (Antonopoulos and Wood, Transactions).

3.4. Ethereum Merkle Tree

Compared to bitcoin, Ethereum uses complex merkle tree structure where each block contains three different merkle trees. Similar to bitcoin, transaction tree is used to find if a block contains a transaction or not. The second tree stores transactions effects or transactions log data known as *receipts*. This tree helps clients in querying information regarding the output of specific transactions. Since every transaction passes through different states until it reaches the final state, it is important to record every state for future querying and validation. The third tree records states of the transactions and the state of accounts that executed them. Clients may query information such as the whether the account exists or not and the balance of the account. The state tree can be used to simulate running the transactions again in a fake temporary block to validate the outputs (Buterin).

3.5. Ethereum Client

Every full node in the network runs Ethereum client software. Full nodes receive transactions propagated though the network. If the transaction is invoking a smart contract, then it is handled by the EVM. The EVM has a stack-based architecture where the *word* size is 256-bit. This size is required to be able to hold Keccak-256 hash and elliptic curve computations. The EVM has its own read only memory (ROM), volatile memory and non-volatile storage (Wood). The client software receives request and export functionalities to EOA and other contracts via an interface known as *JavaScript Object Notation-Remote Procedure Call* (JSON-RPC). JSON is a simple data exchange format written in JavaScript similar to Listing 3.1 while RPC is a client-server communication protocol where the client requests a service from the server and wait until the response is received or timed out. Clients require set of exported functions to communicate with the server known as *Application Programming Interface* (API). For example, Ethereum JSON-RPC API includes *eth_gasPrice* function that provides gas price to the caller. JSON-RPC is not a new technology and it is widely used over HTTP as an interface between webpages and servers (Antonopoulos and Wood, Ethereum Clients).

Listing 3.1

```
{
"jsonrpc":"2.0", // JSON-RPC protocol version
"method":"eth_gasPrice", // API function to answer a request regarding current gas price
"params":[], // extra input parameters if needed
"id":67 // call identifier used for multiple requests
}
```

Similar to bitcoin, each full node has own mempool and a copy of the blockchain. Ethereum node contains another piece of software called *client-process* which is responsible on managing the node transactions by placing them in the mempool, forwarding them to the EVM or updating the blockchain (Infante 32-33).

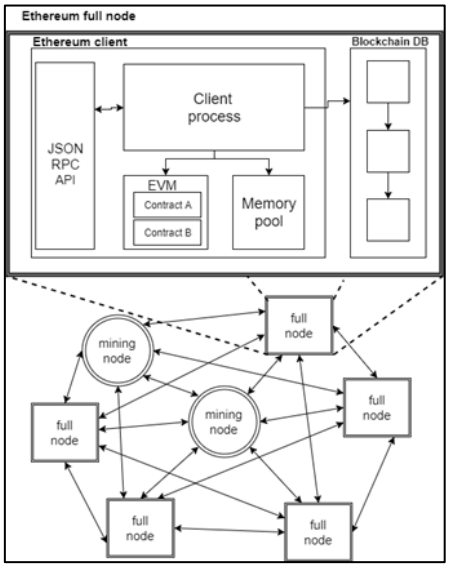

Figure 20. Ethereum Full Node Client (Infante 33)

3.6. Smart Contract

Smart contract is a computer program developed using programming languages and then is compiled into bytecode for the EVM to be able to execute it. The compiled bytecode is then deployed to every full node in the network. The smart contract has a deterministic nature since the output is always the same for all nodes. Smart contracts are immutable and hence cannot be updated to newer versions. The only way to release new version is to delete the older one. Removing the smart contract does not affect the blockchain. All the transactions stored in the blockchain that were created by a smart contract are immutable and stored permanently. To keep the Ethereum clients clean from unnecessary smart contracts, a developer who created a smart contract that is not used anymore would receive an incentive in ETH when deleting it. Removing the smart contract invalidates its address and any future calls by EOA are ignored. Since the EVM was developed specially for Ethereum, developing new programming languages for smart contracts that can be compiled into bytecode was much easier than using popular mature languages such as C language. The most popular programming language is *Solidity* which is very similar to JavaScript and C++ (Antonopoulos and Wood, Smart Contracts).

3.6.1. Solidity Programming Language

Solidity is defined as *contract-oriented language* as an alternative technical term of object-oriented programming language. Solidity has many common principles to other languages such as data types, functions and inheritance. Solidity is complex and under rapid development. Future releases might contain major changes to improve security and performance. Listing 3.2 contains a simple solidity code that creates a smart contract named Zon*Coin* to facilitate transfer of funds or *tokens* (Solidity Docs).

Listing 3.2 (Solidity Docs).

```
pragma solidity >0.4.24;                                    ①

contract ZonCoin {                                          ②
  address public minter;                                    ③
  mapping (address => uint) public balances;                ④

  event Sent(address from, address to, uint amount);        ⑤

  constructor() public {                                    ⑥
    minter = msg.sender;                                    ⑦
  }

  function mint(address receiver, uint amount) public {     ⑧
    if (msg.sender != minter) return;
    balances[receiver] += amount;
  }

  function send(address receiver, uint amount) public {     ⑨
    if (balances[msg.sender] < amount) return;
    balances[msg.sender] -= amount;
    balances[receiver] += amount;
    emit Sent(msg.sender, receiver, amount);                ⑩
  }
}
```

1. Solidity version is defined to assist the compiler. If the version is not defined, this code might fail to compile in future versions.
2. Create a new smart contract named ZonCoin. Solidity is defined as contract-oriented because of using the keyword *contract* instead of *class*.
3. Solidity has a special data type called *address* that can store the 20-bytes of the Ethereum address.
4. Mapping is a key-value data type. In this case, the key is EOA or other contract address and its value is the balance of that particular address. The variable *balances* is marked with *public* keyword and therefore it is accessible from the outside (EOAs or other contracts). Opposed to public, there is *internal* keyword which prevents the access from outside and *private* keyword which prevents access from both outside and from child contracts (inheritance). Finally, there is also *external* keyword which allows the access from outside but prevents contract's functions from calling it.
5. Events are automated mechanism of reporting back to the caller or a listener that an action has occurred. They are also recorded in the transaction logs.
6. Smart contract has a *constructor* that is called only once when the contract is created. Constructors can be used for any required initializations.
7. The local address variable *minter* stores the contract owner address which was originally stored inside the auto-generated *msg* variable that represents the transaction data. Future *msg* variables may come from different EOA users or other contracts. But since the constructor is only called once, the variable *minter* stores the owner address for future usages similar to preventing anyone but the owner from deleting the contract in future.
8. The first function of the contract. It is used to increase (mine) the owner balance. It is marked with public keyword which allows EOAs or other contracts from calling it. However, in this case, the function prevents anyone but the owner from calling it using the *if condition*. The amount is stored in unsigned integer (uint) data type.
9. This function allows sending funds between addresses. First, it checks if the balance of the sender is larger than the amount to be transferred. Next, it starts

deducting the amount from the sender address and adds it to the receiver address.

10. When the send operation completes, the event in point 5 is triggered using *emit* keyword.

There are other essential functions in solidity such as *self-destruct* that can be called by the owner of the contract to remove the smart contract from Ethereum (Solidity Docs).

3.6.2. Tokens

The term token is commonly referred to special purpose coin of insignificant intrinsic value. Probably most mature people have used tokens during their life, for example in amusement park, public laundry or playing old arcade games. Tokens are used usually in Las Vegas casinos where gamblers swap dollars for plastic tokens to be able to participate in games. Unlike money, tokens have limited usages and cannot be used to purchase items or goods and therefore their value is described as insignificant. The concept of token has been borrowed to the Ethereum to represent digital asset ownership, access rights, personal identity, new digital currency, service subscriptions and few others. There are two types of tokens; *fungible* and *non-fungible*. Fungible token can be substituted with other token without loss of value while non-fungible token represents items of unequal unique value and cannot be substituted (Antonopoulos and Wood, Tokens). For example, a gamer playing strategic game decides to purchase a unique castle to defend his kingdom. There is only one item of its kind in the game and highest bid wins it. This digital asset is unique and non-substituted and therefore it is considered *non-fungible* token.

When the tokens represent items like digital assets that only exist in the blockchain, then these tokens are referred to as *intrinsic tokens*. On the contrary, tokens used to represent ownership of houses are called *extrinsic tokens* because the properties exist in the real world and governed by regulation and rules of the country authorities. There have been attempts to migrate some extrinsic tokens to become intrinsic tokens in few

cases. For example, establishing a cooperate companies on blockchain only where users own *share tokens* and can participate in making decisions by voting as shareholder in a decentralized environment. This kind of establishment is known as *decentralized autonomous organization* (DAO) (Antonopoulos and Wood, Tokens).

Figure 21. Casino Tokens (vic).

Tokens are smart contract and in fact Listing 3.2 creates a token that represent a special digital share named ZonCoin. Suppose that Alice is an inventor who has a great idea for a startup called Zon. Alice decided that she is going to establish a company on Ethereum as DAO and the ownership of the startup shares is going to be represented by tokens. There is 100 ZonCoin where each represents one percent of the startup shares. Bob would like to invest in the startup by purchasing 10% of the shares. Alice creates new smart contract called ZonICO. The rule of the second smart contract is to accept Ether from Bob as payment and notify the first smart contract to assign shares to Bob as illustrated in Figure 22 (Antonopoulos and Wood, Tokens).

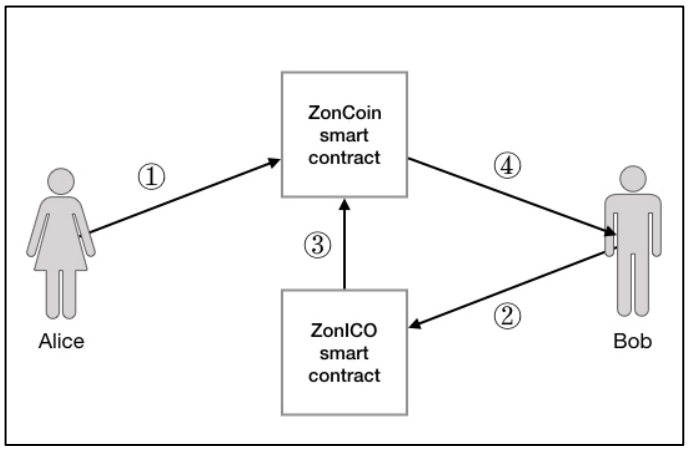

Figure 22. Zon Startup Token.

1. Alice calls *mint* function of the smart contract to issue 100 ZonCoin.
2. Bob buys 10% shares with Ether cryptocurrency.
3. ZonICO smart contract receives the payment from Bob and notifies ZonCoin.
4. ZonCoin assigns 10% shares as tokens to Bob address.

Bob now is a shareholder that is able to participate on the decisions making over decentralized application where shareholders meet online and vote in trustful environment.

If Alice manages to sell enough shares to investors, then she has enough funds to start her business. This kind of funding is known as *crowdfunding* and on Ethereum similar fund rising campaigns are known as *Initial Coin Offering* (ICO). To create a reusability of fungible tokens with other smart contracts, having standard way of creating tokens in essential. The most popular standard is ERC-20 that define an interface for creating smart contracts tokens. The interface includes functions that must be implemented such as *totalSupply* and *balanceOf*. These two functions return total number of tokens and how many tokens is owned by a certain user (Antonopoulos and Wood, Tokens). ZonCoin smart contract in the Listing 3.2 is not practical example of token creations since it does not follow the ERC-20 standard.

3.7. The Anatomy of DApps

The bigger vision of Ethereum system is to have all components of the DApps separated and decentralized. In fact, Ethereum would be responsible for only storing the smart contract of the DApps. Software applications require more than the programming code to work properly. Typically, a web application consists of five core components. The frontend, backend, storage, communications protocol and the name service. The frontend is the webpage that users interact with. Webpages are usually developed using html, CSS and JavaScript. The backend is the application that runs on the server. It receives requests from the frontend, process them and send the results back to the user. The communication protocol between frontend and backend happens over HTTP usually using JSON-RPC API. Web applications might require data storage in the form of SQL database or hard drive to store images, videos and other files. DNS name services exist to offer easier accessibility to web applications using domain names instead of typing the IP-address of the hosts. All first four components can exist on a single machine and therefore, web applications can be referred to as centralized applications.

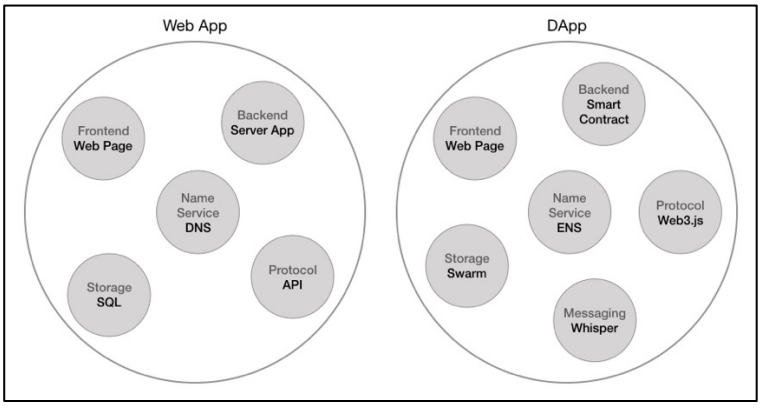

Figure 23. Web App vs DApp Core Components.

The core components of the DApp are very similar. For the webpage, similar programming languages and frameworks can be used. Communication protocol between frontend and backend can occur over HTTP via *web3.js* JavaScript libraries. The

backend for DApp exists on Ethereum nodes and programmed using smart contracts. While it is possible to develop complex backend applications on typical servers, it is quite difficult to do the same using smart contracts due to the gas cost. Therefore, complex logic may exist on a trusted centralized server that communicate with the smart contract. The data storage can be stored outside the Ethereum network on a centralized server or using decentralized P2P solutions such as *Swam*. Unlike web applications, DApps' smart contracts may require communicating with each other to complete tasks. Whisper is a messaging protocol that provides secure P2P communications between DApps (Antonopoulos and Wood, Decentralized Applications (DApps)).

Similar to DNS, Ethereum Name Service (ENS) exist to convert Ethereum addresses to human readable format. ZonCoin smart contract could be referenced as *zoncoin.eth* instead of using 20-byte Ethereum address. ENS is built on smart contracts on the Ethereum network where registering names occur without central authority and through auctions (Infante 256-258).

Swarm is defined as a "distributed storage platform and content distribution service". The storage is distributed over P2P network which makes is resistant to DDoS attack and censorship. Since it does not have single point of failure, data is always available and is fault-tolerant. Once a file is uploaded to swam it cannot be revoked. When file is uploaded to swarm, it is handled by a gateway called *Distributed Preimage Archive* (DPA). DPA send the file to another component called *Chunker* that splits the file into chunks of maximum 4-byte size. Each chunk is hashed and stored in a merkle tree data structure. The root of the file's merkle tree (known as key root) is calculated and is send back to the DPA which in turn sends it back to the uploader. The uploader can use the key root hash to download the file from swam network. The interesting part of swarm that the file's chunks are distributed over many swam nodes instead of storing the whole file. Swam communicates with each other via protocol call *bzz* (Infante 256-270). Each swam node runs SWAP accounting protocol that awards the node with micro payments for the bandwidth used for sharing chunks. Popular contents can be replicated across the nodes since serving them have higher incentives. Some files might be uploaded and rarely requested. This creates a burden for nodes to store the unpopular chucks for

longer period. Therefore, nodes could receive storage incentives for not removing these unpopular chunks (Trón, Fischer and Nagy). Alternative to swarm, there is another solution for decentralized storage called *InterPlanetary File System* (IPFS). IPFS splits the uploaded file into blocks of maximum size of 256-byte. Each block is identified with cryptographic hash and the nodes store only the blocks they are interested in. IPFS provides a mechanism to remove duplications and offers tracking system for files versions. Files can be identified using human readable format using IPNS naming system (ipfs.io).

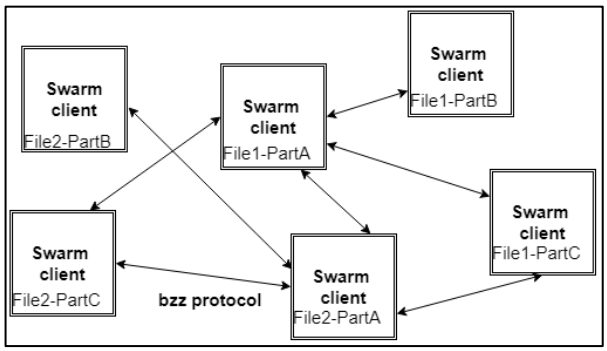

Figure 24. Chunks Distribution Over Swam Nodes (Infante 266).

Ethereum Smart contracts are not allowed to fetch data from external sources to protect the blockchain integrity. Also, from technical point of view, since the same smart contract is replicated across the network, it is difficult to coordinate and choose a single smart contract that is responsible for communicating with the external internet. The solution is to create a trusted intermediary component responsible for fetching and validating external data sources. Ethereum proposed a solution named *Oracle* to fetch data from external sources and validate that the data source before returning the authenticated data to the smart contract. The smart contract requests authenticity proofs from oracle when requesting data. Oracle service provider (Oraclize) offers different authenticity proofs such as TLSNotary (oraclize.it). TLSNotary concept is about proofing to a user called “auditor” by another user called “auditee” that data is valid through using cryptographic solutions over *Transport Layer Security* (TLS) protocol (tlsnotary.org).

3.7.1. Truffle Suite

Every software development pipeline involves repetitive tasks and initial codes (boilerplate code) that must be included in every project. Repetitive tasks include, creating new project, selecting the environment settings and configurations, compiling, testing and deploying software to production. DApps development, like any other applications development, require tools and frameworks to accelerate development cycles and make the life of the programmers much easier. Therefore, Consensys (the company behind Ethereum) released *Truffle* development suite which is made of three components: Truffle, Ganache, and Drizzle.

Truffle is the core framework of the suite. New projects can be created quickly using a single command. It offers the possibility for writing automated testing using Solidity or JavaScript. Automated testing, also known as *Unit Testing*, become fundamental part of every software development cycle. Truffle comes with tools for debugging smart contracts and deploying them quickly to Ethereum network (truffleframework.com).

Using Ganache, it is possible to create personal blockchain. Personal blockchain offers the possibility of emulating the whole Ethereum network on a single local development machine. Smart contracts can be deployed locally, and transactions can be monitored as if they are running on real Ethereum network. As mentioned earlier, DApps require frontend component that can be developed using web technologies. Drizzle framework is a collection of frontend libraries based on the popular React and Redux web frameworks. Using Drizzle, developers can craft quick frontends that synchronize to web3.js easily (truffleframework.com).

Figure 25. Truffle Suite (truffleframework.com).

3.7.2. DApps Use Cases

The Swiss town of Zug, which is known as the crypto valley, was the first city to conduct a non-binding municipal electronic voting over the blockchain using mobile devices. The trail involved 240 citizens who had access to the e-voting system. The town announced the success of the trail where only 72 citizens participated in the vote (swissinfo.ch). The town developed blockchain identity system known as eID using uPort protocol. The uPort consist of smart contracts being developed as digital identity layer over the Ethereum network where DApps can have access to it (uport.me).

CryptoKitties was one of the first blockchain games and digital assets ownership. Players collect and breed unique digital cats. It has been reported that one kitten was sold for equivalent to $114,000 and players have spent more than $6.7 millions (Cheng).

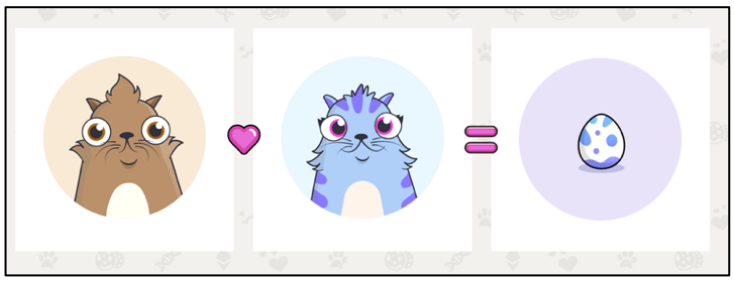

Figure 26. CryptoKitties DApp (cryptokitties.co).

One of the most used DApp is Idex decentralized exchange. Decentralized exchange enables users to trade ERC-20 tokens over Ethereum without central authority. Information and insights about all existing DApps can be viewed on dappradar.com website (Floyd).

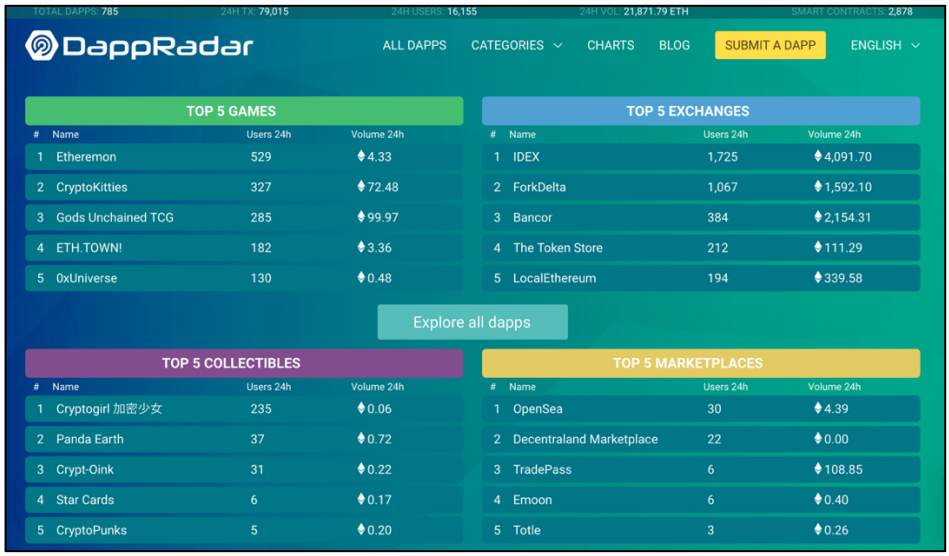

Figure 27. DappRadar.com Website.

3.8. Critiques

The main major technical issues for Ethereum are scalability and energy consumption. Ethereum can process roughly 15 transactions per second which is a big drawback for the dream of alternative internet (Blenkinsop). In fact, Ethereum traffic slowed down when CryptoKitties was launched forcing other transactions to remain in the mempool for long time. CryptoKitties consumed alone roughly 20% of all Ethereum computation power (Hertig). Scalability problem has been discussed over long time and new startup companies are trying to provide solutions. The Liquidity network[1] offers off-chain

[1] https://liquidity.network/

payment protocol while the Lightning Network[1] offer a solution to create a payment channels between users without recording every transaction in the blockchain.

Knowing these limitations, Ethereum could have focused on providing solutions to the current internet rather than to reinvent it. People would use Ethereum side by side with the internet for applications that require trustful environment. Games and social applications are not top priority for people and companies interested in blockchain. Ethereum would have faster adoption if it was called something like the trustful internet rather than the Web 3.0. People could use DApps to purchase items or trade goods. Having a decentralized social network may sounds fancy to people considered about privacy but do the majority of people around the globe really care?

While Proof-of-Work is a smart consensus protocol that protect the blockchain network from DDoS attack and scammers, it consumes a lot of energy and slows down transactions. Slow transactions might be acceptable for bitcoin, but certainly is not for Ethereum. Ethereum is proposing new consensus protocol called *Proof of Stake* (PoS). The alternative PoS consensus proposes that miners must send a deposit of cryptocurrency (Ether) to be able participate in mining process. If a miner was caught cheating or attacking the network, that deposit will be panelized. There are many proposals on how the consensus protocol should be implemented. Ethereum upcoming implementation is named *Casper*. The first version of Casper was released recently under the name of the *Friendly Finality Gadget* (FFG) (Ray, Proof of Stake FAQs).

Implementing PoS might open the door for miner with higher stakes to control the network. It might save energy, but it does not solve the problem of having a super *crypropower* discussed before. Aren't large cooperate controlling the internet traffic today because the they are rich and have large resources?

A Germany company (Slock.it) launched a popular DAO on April 2016. The DAO raised over $150 millions from crowdfunding ICO campaign run on smart contract. The DAO smart contract had a bug that allowed a hacker to steal around $50 millions from

[1] https://lightning.network/

the funds. The only way to recover the stolen funds was to split Ethereum blockchain by performing a hard fork. The dream of moving cooperate into blockchain faced the reality (Jentzsch).

Since blockchain is still immature technology for establishing businesses over the internet, non-technical businessmen would not see any propose of establishing online companies using DAO. If DAO offers transparent voting, why not just establishing regular company and then perform voting over the Ethereum?

A new research study published by Satis Group found that around 78% of token ICOs were scam projects with no plans for investing the funds in reality. More than $1.3 billions were lost in only three major fraudulent funds (DeLisle). One of the biggest ICO scam was a decentralized banking platform named Arisebank. United States regulators shut Arisebank ICO after raising $600 millions funds from investors (Rapier).

Unfortunately, criminals have discovered ways to steal investors' money due to the large hype that surrounds the blockchain technology and because of Ethereum promoting big names such as the next internet (Web 3.0) or the world computer. This risk could have been reduced if there were fair articles and conferences that explain the limitations and risks for investors which would save losing funds in scam projects that could have been invested wisely.

Two more problems have been raised recently. The first one occurred when a new Chinese exchange service *FCoin* launched its DApp using new business model resulting in dramatic increase in gas price. As a result, the network suffered congestions and the average gas price spiked to what was equivalent to $3.2 per transaction (Bluetower).

Again, this problem is related to scalability limitation of the Ethereum. For users to maintain interests in using Ethereum, it is recommended to lower gas prices or eliminate gas completely at this stage of development and reward miners with new Ethers and mining fees until the scaling issue is resolved.

The second problem was regarding the growth of the blockchain data. It was reported by etherscan.io website that the current data size about 93 GB. While this size may sound normal today, it rises concerns for future. How large the blockchain data can grow since it has to be replicated and synced on thousands of times on every full node on the network?

Solidity programming language has been criticized for being vulnerable to several attacks. The list of known attacks is available on the official website. An alternative language called Vyper, similar to python, has been suggested to improve the security of smart contracts. Currently, Vyper programming language is still in the experimental stage. At this stage and due to this uncertainty regarding tools and programming languages, it is might be difficult to attract programmers to develop DApps. Learning new tools and programming languages require efforts and time.

Another serious issue may occur when organizations or countries relies on third parties to assist them in providing blockchain solutions. Blockchain is about eliminating third parties in the transactions to create trust. It has been reported that government of Sierra Leone run election over the blockchain with the help of Agora, a Swiss startup company. It appeared later that Agora was monitoring the election and the database they were using was not run on blockchain. Worse, Agora results mismatched with the official ones (Economist).

This proves that, any country or organization interested in blockchain is required to develop their own solution. For Ethereum to find success and higher adoption, developing Ethereum DApps should be simplified as much as possible.

Finally, similar to the mobile apps, the interest of using DApps may fade quickly. When smart phones launched years ago, people were very excited about downloading all kind of applications and games. Nowadays, the interest has faded and in fact the popular CryptoKitties has already seen it. The number of visitor to CryptoKitties DApp has decreased by 96% according to a report published recently (Roberts).

4. THE HYPERLEDGER PROJECT

4.1. Introduction

The rise of blockchain technology has attracted many large companies to invest in utilizing the blockchain to serve the enterprise. There are strong demands for creating trustful systems without central authorities in many fields from finance to logistics and trade. Instead of working separately, many large companies decided to collaborate under the umbrella of Linux Foundation. As a result, Hyperledger open source project was launched in 2015. The project has attracted hundreds of organizations including IBM and Airbus. The purpose of Hyperledger project not to create single blockchain solution, but rather to bring users, developers and vendors together to create the future of the enterprise blockchain. Unlike Ethereum, Hyperledger blockchain solutions are mostly permissioned blockchain. They deliver solution to create private blockchain for certain organization or impose specific restrictions on public users. Users may have different rules according to their positions at the organization they work for. Currently, there are five open source blockchain solutions developed by different companies and number of tools to accelerate development as illustrated in Figure 28. Each of these solutions follow a standard design philosophy. First of all, they must have a modular design where components can be reused across different projects to help the developers to perform different experiments. Security is the second design philosophy where each solution has to follow Linux Foundation best practices for creating highly secure systems. To keep all projects under one greenhouse, smart contracts and decentralized applications must have high degree of interoperability to work on different projects. The fourth design requirement distinguishes Hyperledger from bitcoin and Ethereum by not having own cryptocurrency and only allowing the usage of tokens to manage digital assets. Finally, all solutions must have API to support applications developers and to make the solution accessible via external systems (Group 3-11).

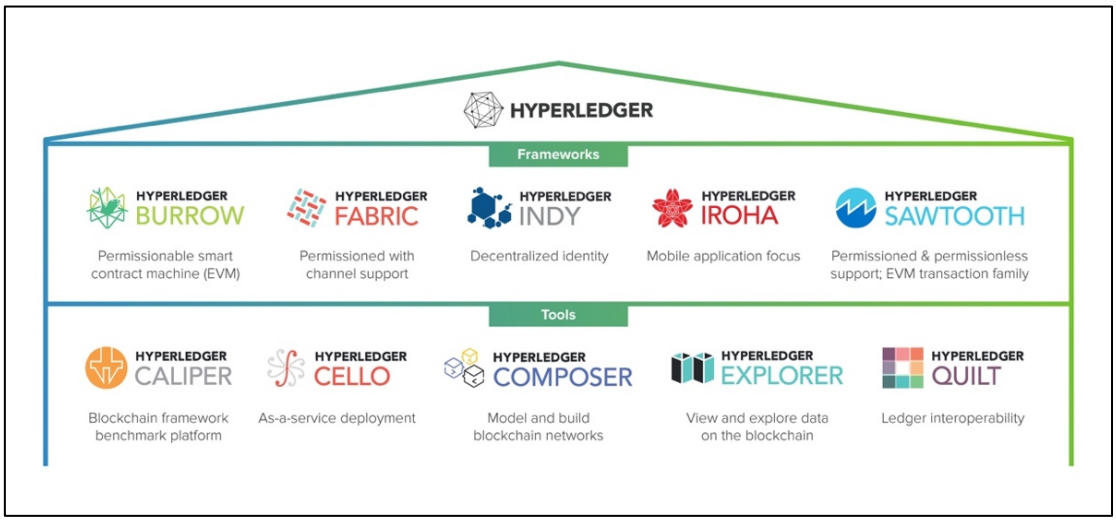

Figure 28. Hyperledger Project Greenhouse

Hyperledger Indy is a distributed ledger for creating digital identities similar to Ethereum uPort. A single user can have one digital identity that can be used in all organizations that implement Indy in their blockchain. Hyperledger Burrow has been designed similar to Ethereum client software with one major exception; smart contracts are executed in permissioned EVM. Burrow offers businesses the ability to run Ethereum smart contracts on private confidential blockchain by limiting the access to users with permissions. Hyperledger Iroha is a simple general purpose blockchain with main focus on creating and managing digital assets. Iroha provides Java, Python, JavaScript, iOS and Android SDKs for quickly developing decentralized applications. Iroha objective is to assist organizations such as banks in adopting blockchain solutions simply and quickly. Hyperledger Sawtooth is customizable blockchain for business. It offers permissioned or permissionless blockchain and the ability to run Ethereum smart contracts with the integration of Burrow. It offers own consensus protocol known as *Proof of Elapsed Time* (PoET). Hyperledger Sawtooth consensus is dynamic and changeable according to the organization needs. Applications SDKs are available in several programming languages including Python, JavaScript, Go and Rust. Finally, Hyperledger Fabric is similar to Hyperledger Sawtooth. Fabric has own smart contract known as *chaincode*. Components and consensus are pluggable according to the organization requirements. Unlike Sawtooth, Fabric provides only permissioned blockchain model (Group 22-27).

Hyperledger tools aim toward simplifying working and managing the Hyperledger frameworks. For example, installing, deploying, developing and providing easier migration to different blockchain. Hyperledger Cello defines itself as a blockchain operating system. It provides automated functions for managing and monitoring the blockchain. Organization using Cello can install customized consensus rules, deploy the blockchain to Docker or Kubernetes cloud containers, and provides support for Swarm decentralized storage. It is possible to build a *blockchain as a service* (BaaS) and lease the service for smaller companies. Currently, it supports Fabric with plans to support Sawtooth in the near future (Cello Github).

Hyperledger Composer accelerates the development blockchain application by providing businesses tools to develop case studies and deploy them quickly to the blockchain. It offers the possibility of integrating blockchain with current systems via API interfaces. The Hyperledger Explorer project provides web application for exploring all kind blockchain components such as the blocks, nodes, network status and the transactions. At the moment, both Composer and Explorer support only Hyperledger Fabric. Hyperledger Quilt aims toward providing interoperability between different blockchain solutions through Interledger Protocol (ILP). ILP is a payment protocol that enables value transfer between distributed and non-distributed ledgers. Using Quilt, financial institutions and supply chain implementing different blockchain solutions will be able to connect and communicate with each other. The last tool is Hyperledger Caliper which provide benchmark tools for measuring the blockchain performance. For example, it is possible to measure how many transactions per second the blockchain network is able to execute. The current release of Caliper supports Fabric, Sawtooth and Iroha (Group 27-29).

Hyperledger project opens the possibility for organizations to choose the solution that fits with their requirements while enabling all the frameworks to communicate with each other as if they all belong to the same ecosystem.

4.2. Hyperledger Sawtooth

While all Hyperledger projects share the same design philosophy, they have different architecture and technical concepts. Hyperledger Sawtooth project has been chosen to be covered in more details since Intel Cooperation says it is production ready and already being tested by some large companies like T-Mobile to develop business-ready blockchain solutions. T-Mobile is developing *Next Identity Platform* utilizing Sawtooth for its own subscribers' identities and assets management (Moos). Sawtooth node is referred to as a validator node because there is no mining competition in Hyperledger and no rewards in cryptocurrencies. Validator node runs the Sawtooth client software with full blockchain copy and forms P2P network with other validator nodes. Clients applications communicate with the node via REST API using HTTP protocol and JSON format. Interconnect is another component for establishing alternative communication channel over gossip protocol such as TCP or UDP. The validator node maintains a global *state* of all transactions using a single *radix merkle tree*. Radix is an address format used for representing root and leafs addresses. The full address length is 35 bytes and is represented in hexadecimal format. The first 3 bytes of the radix address represent a namespace to separate transactions according to their types (families) (Sawtooth Docs). Transaction processors, consensus and validator will be discussed shortly.

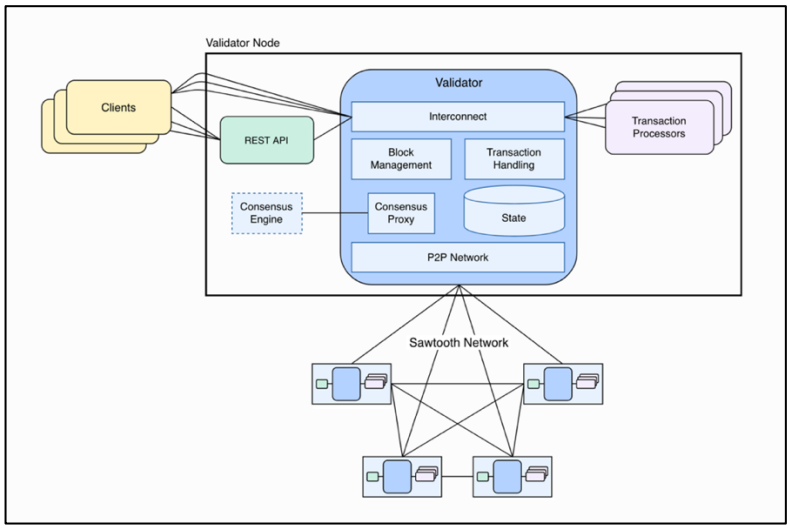

Figure 29. Hyperledger Sawtooth Network Overview (Sawtooth Docs - Introduction).

4.2.1. Transactions

Sawtooth transactions are customizable according to the organization requirements. Transactions can be of a complex type similar to Ethereum smart contract or of a very simple type. One blockchain implementation may run different types of transactions in parallel. Transactions types are referred to as *transactions families* and the current release of Sawtooth provides a predefined set of families including simple *IntegerKey* family that allows only three operations: increment, decrement and set. Organizations implementing Sawtooth should be aware that choosing the right family is important factor in the terms of versatility and risk (Middleton). Bitcoin uses limited transaction specifications to reduce the risk while Ethereum adopted complex smart contract specifications for the sake of versatility. Since the Sawtooth is a modular blockchain, organizations may also develop their own set of families. Developing custom transactions families requires programming *transaction processors* using provided SDKs which are available in different programming languages including Java, Python, JavaScript, Go, C++ and Rust (Sawtooth Docs - Introduction).

Since transactions may have different structures and belong to different families, it is possible that some transactions depend on other transactions. Assuming three transactions A, B, and C have cyclic dependency where C depends on B, B depends on A and A depends on C. Propagating each of these transactions alone to the network would create complexity. None of them can be validated and added to the blockchain because of the cyclic dependency. Transaction A requires that transaction C already validated and inserted to the blockchain while transaction B expects the same from transaction A. Therefore, to solve this complexity, transactions in Sawtooth are grouped inside a *batch* before being propagated for validation. All the three transactions A, B and C could exist in the same batch and are treated and validated as a single unit. If any of the transactions inside the batch is invalid, the whole batch is discarded. Transactions dependency can occur within the same transaction family or different families. Imagine the scenario where different applications work together to complete a single task and each application has own custom transaction family (Sawtooth Docs Transactions and Batches).

Transaction structure contains three fields; *header* (serialized into bytes), *header signature* and *payload*. The payload is the transaction data or functions that when are executed change the state of the transaction. The payload contents depend on the transaction family. For example, IntegerKey family uses the following contents represented in COBR format (similar to JSON format):

Listing 4.1. (Sawtooth Docs - IntegerKey Transaction Family)

```
cbor.dumps({
   'Verb': 'set',
   'Name': 'foo',
   'Value': 1234,
})
```

The payload contains three variables. The *verb* is used to set the action that the transaction will execute. In this case, it is either increment (inc), decrement (dec) or set. The *name* variable refers to the record to be modified. Finally, the *value* variable contains the integer value (Sawtooth Docs - IntegerKey Transaction Family). The *header signature* is the result of signing the *transaction header* with the user private key. The transaction header field contains set of fields as illustrated in Figure 30.

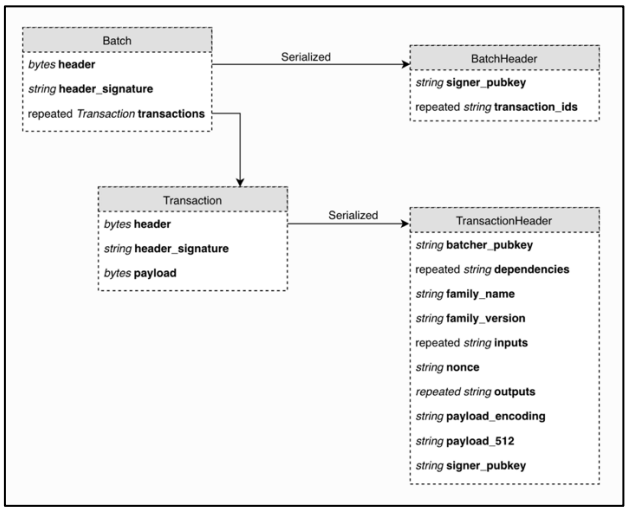

Figure 30. Sawtooth Transaction (Sawtooth Docs - Transactions and Batches)

Transactions header contains transaction family name, version and a nonce that makes the transaction unique. It contains encoded payload content using SHA-512 hash function. The singer public key is included to confirm the header signature. Some transactions may depend upon prior transactions which are not included in the same batch. In this case, *dependencies* field is used to link the transaction to previous external transactions. The client public key that create the batch is included in the header to prevent the transaction from being reused in different batch. Similar to Ethereum, transaction passes through different states until it reaches final state where it is recorded into the blockchain. The last two fields *inputs* and *outputs* are used to assist parallel scheduling operations. Inputs field contains transaction states addresses to read from and *outputs* field contains state addresses to update. The contents of *inputs* and *outputs* depend on the transaction family. For example, IntegerKey family uses the address of the *name* field in both the inputs and the outputs. The transaction is then added to a batch which contains a header and header signature fields as well. Both headers are hashed using SHA-256 and then signed with private key using ECDSA to generate 64-byte digital signature (Sawtooth Docs - Transactions and Batches).

4.3. PoET Consensus

Hyperledger Sawtooth project is contributed and developed by Intel Cooperation. The company developed an alternative consensus based on PoW concept but without the huge energy consumptions. The new consensus known as *Proof of Elapsed Time* (PoET) is based on simple idea. Every node generates random time and wait until the time expires. The node with the shortest time is the winner and therefore owns the right to create the new block. The concept is very similar to PoW where the first node (shortest time) to solve the cryptographic puzzle is announced as the winner. PoET consensus requires Intel SGX technology for protection against cheating or malicious attack (Sawtooth Docs - PoET 1.0 Specification).

Intel SGX creates hardware protected memory execution environment called *enclave*. While the enclave is created by the software, the execution of the program that runs

inside the enclave is inaccessible externally by the operating system or the software that initialized the enclave. The integrity of the hardware and the program can be attested externally using public key cryptography (Intel SGX).

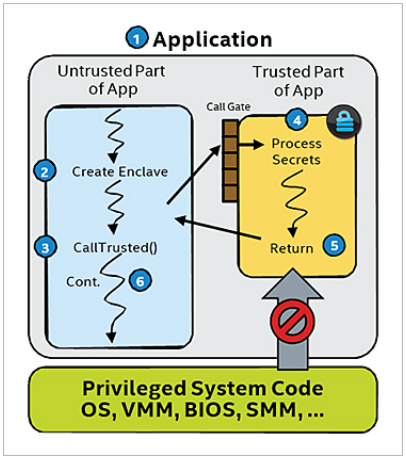

Figure 31. Intel SGX Runtime Execution (Intel SGX).

Several research papers have been published on Intel SGX security model including a research paper by Ohio State University discovered that Intel SGX is vulnerable to sniffing attack. The SgxPectre attack can be used to steal private keys as described in the paper "we have applied SgxPectre Attacks to steal seal keys and attestation keys from Intel signed quoting enclaves. The seal key can be used to decrypt sealed storage outside the enclaves, and the attestation key can be used to forge attestation signatures" (Chen, Chen and Xiao).

4.4. Validator's Journal

The components that are responsible for creating or validating new blocks and managing the blockchain data are called the *journal*. The first component is the *Completer* which is responsible for receiving both the transactions batches and new proposed blocks (created by other winner nodes). When a new proposed block arrives, it contains header information such as batches Ids. The completer must check if all

batches Ids are included in the new proposed block have been already received and must resolve their dependencies as well as confirming that the parent block is stored in the blockchain before considering the new block as complete and then forward it to *ChainController* for validation. Batches are considered complete when their dependencies are resolved and then forwarded to *BlockPublisher* to be included in a new block (Sawtooth Docs - Journal).

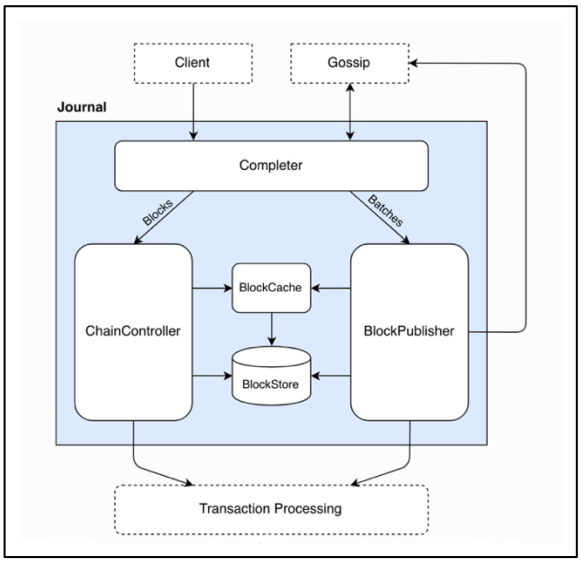

Figure 32. Sawtooth Journal (Sawtooth Docs - Journal).

The ChainController is responsible for validating the new proposed block and inserting it to the blockchain. It is also responsible for resolving forks if they occur. The BlockPublisher is responsible for creating candidate block but may only insert it to the blockchain or propagate it to the other nodes when it receives the instruction from the consensus protocol that the host node was the winner. The blockchain in Sawtooth is called the *BlockStore*. Sawtooth introduces another memory-only blockchain named *BlockCache* that contains recent and relevant blocks and their states (valid, invalid or unknown). Similar to memory cache in computers, BlockCache provides high speed access for the validator. Blocks can be removed from the BlockCache if they have not been accessed or queried for long time. If the host node is restarted, all the contents of the BlockCache are cleared (Sawtooth Docs - Journal).

4.5. Network and Transactions Permissions

Sawtooth supports two types of permissions. The first one is set to restrict who is allowed to submit transactions (or batches) to the network and the second one is set to restrict the validator nodes that are allowed to join the network. Based on the permissions enforced, the blockchain network can be classified into three types: public, private and consortium. Public network allows all transactions and batches from any client and allows any node to join the network and become a validator. Public network is similar to Bitcoin and Ethereum networks. Private network restricts accepting transactions and batches unless the public keys used in signing process are whitelisted. The same restrictions apply on accepting new node joining the validator network. The consortium network allows any client to submit transactions and batches while restricts nodes joining the validator network (Sawtooth Docs - Permissioning Design).

4.6. The Food Journey

Supply chain companies are among the top interested companies in blockchain technology. Blockchain solutions offer traceability, immutability and eliminate the lack of trust that exist in current working environments. Food safety is among the top priorities for retailers and logistics companies. Tracing back every manufacturing step of food before they arrive to customers' tables were always a difficult to achieve due to a chain of different actors involved in the process. Recently, the giant Finnish S-Group company launched a blockchain solution based on IBM Food Trust program built on Hyperledger Fabric for tracing fishes' journeys from lakes and oceans to customers' tables. The solution named *Pike-Perch Radar* provides the customers the ability to check all the information about the fish they are buying using QR code including the site it originated from (Lehikoinen). Intel is also developing a similar solution using IoT sensors and Hyperledger Sawtooth as illustrated in Figure 33.

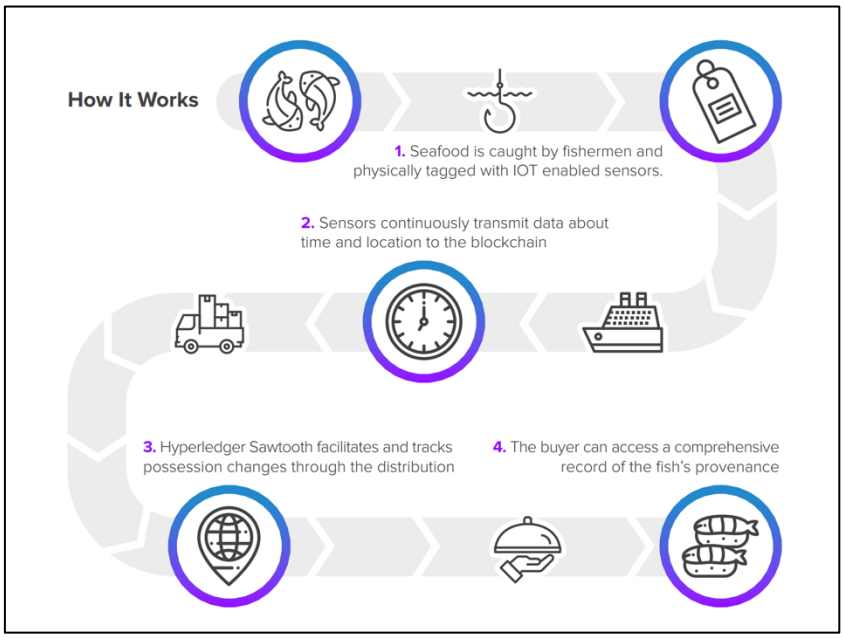

Figure 33. Fish Journey from Ocean to Table (Group 21)

4.7. Critiques

It is not visible to me how non-technical organization will manage to run their own private blockchain without relying on third party. It would be hard for a supply chain or other non-technical organization deploying and managing hundreds or thousands of cloud containers to create own private blockchain network. The solution developed by S-Group is built on top of IBM blockchain business service for food trust. The initial goal of bitcoin innovation was to eliminate third party from trade. But it seems with Hyperledger projects, this is not possible for most of the small and middle size companies. Organizations interested in blockchain will become customers for large cooperate such as IBM. The blockchain service providers will become key player in organizations businesses which may result in future in increasing fees or hiding hardware vulnerabilities and system breaches to protect their business.

Relying on a company to provide blockchain solutions to governments might be riskier. The Australian government signed one billion AUD deal with IBM. The deal includes providing the government with blockchain solutions based on IBM Hyperledger Fabric project (Burton). IBM has been offering blockchain solutions that run on their own

cloud service. While the details are unknown yet, it is unreasonable if government data are hosted and managed by a company on their cloud service.

Unlike Ethereum that found interests from individual developers for the concepts of DApps, the Hyperledger projects might not be able to attract large community of individual developers. Individual developers have been a key success in web and mobile platform in the past. Their contributions have resulted in millions of applications. It could be argued that the main reason why Hyperledger will face difficulties on attracting developers is because the project focuses on permissioned private networks and contributors like IBM and Intel are utilizing the Hyperledger solutions to run on their own hardware and cloud services. Lack of developers will create difficulties for organizations and governments that aim toward running own network and implementing own solutions based on Hyperledger projects.

Hyperledger projects, the general customized blockchain solution, might face strong competition from other solutions that aim toward solving practical problems. For example, Google is working on blockchain and AI solution for the health industry where patients' records will be recorded and handled using the blockchain (Metz). Unity, the game engine leader, has formed partnership with Enjin to provide blockchain gaming digital assets solution (enjincoin.io). Corda is another blockchain solution for business with focus on financial institutions (corda.net). Microsoft is developing CoCo Framework to accelerate developing blockchain solutions regardless of the core technology, whether it is Hyperledger Sawtooth, Ethereum, Corda or others. CoCo Framework may dominate the blockchain development instead of Hyperledger SDKs and tools such as Hyperledger Composer (Russinovich).

Hyperledger contributors are required to accelerate the development and improve the tools to earn the leadership in the blockchain for the enterprise. Probably there should be more specific tools and frameworks for different purposes. It would be easier for companies to understand that this particular solution is suitable for the logistic industry and another one for finance. It also appears that Fabric and Sawtooth are more competing against each other rather than collaborating.

5. Conclusion

Blockchain is an innovative technology but might not be suitable for all organizations or projects. It is not a magic technology that is going to solve every technical problem exists as been promoted sometimes. It is very suitable for systems where different actors participate in the process and no trust can be established between them. The technology is under rapid research and development by many large cooperates and it might have a bright future in supply chain, finance and few other industries. However, the future of digital cryptocurrencies where the blockchain technology originated from is unclear. Bitcoin might survive but decline to smaller audience or it might become much more popular than it is today.

The first chapter of the thesis discussed what is Bitcoin and why is required as an alternative monetary system. Then, the Bitcoin core technology and how the Bitcoin network works were explained. Since all blockchain solutions are based on Bitcoin concepts, explaining the concepts technically were essential for the following chapters. Bitcoin, like any other technology, has limitations and risks. Possible attacks and their potential damages on future were introduced. In the last section of the chapter, Bitcoin received criticism for issues such as power consumptions, lack of representation, foggy future regarding regulations, lack of research and development, and the risk of quantum computing imposing on cryptography.

No doubt that Bitcoin has caused disruptions in the financial sectors specially with the wide adoption by individuals around the globe. However, the technical concepts used to create Bitcoin have opened doors for new innovations in other fields of technology. Bitcoin technology establishes trust in a non-trustful anonymous environment and that has been required by different industries. The press has adopted the technical term “blockchain” to refer to upcoming technologies based on Bitcoin immutable ledger and the consensus protocol driven by the target of establishing trust between participants without having a central authority.

The second chapter covered Ethereum blockchain and the concepts of decentralized applications known as DApps. Ethereum extends Bitcoin technologies with new concepts and techniques. Similarities and differences between both solutions were covered as well as how DApps ecosystem works and what are the programming languages and tools available to develop DApps. Finally, Ethereum received criticism for serious issues including its limited capabilities comparing to its big promises, scalability, and for being hub for ICO scammers.

Ethereum invention carried big promises about the future of the internet and how decentralized network can create one big giant world computer. These promises faced reality and we are still far away from achieving its dreams. These promises have had doubts from the beginning by some people but Ethereum believers say that our internet had similar doubts when it was launched. Probably in three years, it will easier to witness if the promises of the Ethereum will be fulfilled or not.

The third chapter covered private blockchain solutions for the enterprise contributed by the Hyperledger project. Different solutions, frameworks and tools are presented briefly. The second part of the chapter explained Hyperledger Sawtooth framework; one of the project under the Hyperledger umbrella. Technical details are covered and compared against Bitcoin and Ethereum and new concepts are presented. Finally, Hyperledger received criticism regarding few issues such as the difficulty to manage by organizations without relying on middlemen and for being utilized by contributors to run on their own cloud, software and hardware.

Private blockchain for the enterprise might appear as rational requirements where many companies are willing to invest in Hyperledger project to solve their long-time issues. While Hyperledger project aims toward the enterprise first, the open source technology contributed by Hyperledger projects opened another view on blockchain innovations. Hyperledger might find wider adoption by the enterprise in the next few years when frameworks and tools are developed further, and experimental projects record success.

The initial plan was to implement a case study using one of the solutions discussed in this thesis. The case study would discuss implementing blockchain solution for education to create an authenticated certificate system and to accelerate admissions for postgraduate students between partner universities. However, due to uncertainty about the tools and frameworks available today, rapid changes, and lack of technical and programming resources, the case study has been postponed for future work.

Finally, with the current rapid changes of Blockchain technology, it appears that in the next two years; new tools and frameworks will be available and current ones will be improved. More resources, books, tutorials will be published, and more universities will start offering blockchain courses. Blockchain will be thought from different perspectives including technology, managements and law. Blockchain solutions will be implemented by some industries and will be completely forgotten by many others.

6. LIST OF REFERENCES

2018. *bitcoinpaperwallet.* 21 July 2018. <https://bitcoinpaperwallet.com/>.

n.d. August 2018. <https://www.cryptokitties.co/>.

Acheson, Noelle. *coindesk*. 29 January 2018. 22 July 2018. <https://www.coindesk.com/information/paper-wallet-tutorial/>.

Anton Badev, Matthew Chen. "Bitcoin: Technical Background and Data Analysis." *IDEAS Working Paper Series from RePEc* (2014).

Antonopoulos, Andreas M. and Gavin Wood. "Transactions." *Mastering Ethereum*. O'Reilly, 2018.

Antonopoulos, Andreas M. and Gavin Wood. "Decentralized Applications (DApps)." *Mastering Ethereum*. O'Reilly Media, 2018.

Antonopoulos, Andreas M. and Gavin Wood. "Ethereum Clients." *Mastering Ethereum*. O'Reilly Media., 2018.

Antonopoulos, Andreas M. and Gavin Wood. "Introduction." *Mastering Ethereum*. O'Reilly Media, 2018.

Antonopoulos, Andreas M. and Gavin Wood. "Keys and Addresses." *Mastering Ethereum* . O'Reilly Media, 2018.

Antonopoulos, Andreas M. and Gavin Wood. "Smart Contracts." *Mastering Ethereum*. O'Reilly Media, 2018.

Antonopoulos, Andreas M. and Gavin Wood. "Tokens." *Mastering Ethereum*. O'Reilly Media, 2018.

Antonopoulos, Andreas M. "Mastering Bitcoin." 2017. *https://github.com/bitcoinbook/.* July 2018.

Bitcoin.org. *Choose Your Wallet*. 2018. 22 July 2018. <https://bitcoin.org/en/choose-your-wallet>.

BitcoinWiki. *Hardware wallet*. 2018. <https://en.bitcoin.it/wiki/Hardware_wallet>.

—. *Proof of work*. 15 May 2016. 6 August 2018. <https://en.bitcoin.it/wiki/Proof_of_work>.

Blenkinsop, Connor. *Blockchain's Scaling Problem, Explained*. 22 August 2018. August 2018. <https://cointelegraph.com/explained/blockchains-scaling-problem-explained>.

Bluetower, Danielys. *Fcoin Tips Ethereum's Gas Crisis*. 7 July 2018. August 2018. <https://elevenews.com/2018/07/07/fcoin-tips-ethereums-gas-crisis/>.

Burton, Tom. *Australian government and the Big Blue mint $1 billion advanced technologies deal*. 5 July 2018. September 2018. <https://www.themandarin.com.au/95308-australian-government-and-big-blue-mint-1-billion-advanced-technologies-deal/>.

Buterin, Vitalik. *Merkling in Ethereum*. 15 November 2015. September 2018. <https://blog.ethereum.org/2015/11/15/merkling-in-ethereum/>.

Castor, Amy. *A Short Guide to Bitcoin Forks*. 27 March 2017. 6 August 2018. <https://www.coindesk.com/short-guide-bitcoin-forks-explained/>.

Cello Github. n.d. August 2018. <https://github.com/hyperledger/cello>.

Chen, Guoxing, et al. "SgxPectre Attacks: Stealing Intel Secrets from SGX Enclaves via Speculative Execution." 2018.

Cheng, Evelyn. *Meet CryptoKitties, the $100,000 digital beanie babies epitomizing the cryptocurrency mania*. 6 December 2017. August 2018. <https://www.cnbc.com/2017/12/06/meet-cryptokitties-the-new-digital-beanie-babies-selling-for-100k.html>.

CoinDesk. *Hard Fork vs Soft Fork*. 16 March 2018. 7 August 2018. <https://www.coindesk.com/information/hard-fork-vs-soft-fork/>.

corda.net. n.d. September 2018. <https://www.corda.net/>.

DeLisle, Bill. *SATIS Group Report: '78% of ICOs are Scams'*. 17 July 2018. August 2018. <https://cryptoslate.com/satis-group-report-78-of-icos-are-scams/>.

Digiconomist. 4 August 2018. 4 August 2018. <https://digiconomist.net/bitcoin-energy-consumption>.

Economist, The. *The promise of the blockchain technology*. 1 September 2018. September 2018. <https://www.economist.com/technology-quarterly/2018/09/01/the-promise-of-the-blockchain-technology>.

enjincoin.io. *enjincoin news*. 21 March 2018. September 2018. <https://blog.enjincoin.io/press-release-enjin-and-unity-technologies-advance-true-item-ownership-through-asset-store-2ad7b4ed9f4a>.

ethgasstation.info. 12 August 2018. 12 August 2018.

Floyd, David. *The Top 5 Ethereum Dapps By Daily Active Users*. 10 June 2018. 23 August. <https://www.coindesk.com/ico-founders-will-wait-7-more-years-for-crypto-paydays/>.

Group, The Hyperledger White Paper Working. *An Introduction to Hyperledger*. Hyperledger Publications, 2018.

Hertig, Alyssa. *Loveable Digital Kittens Are Clogging Ethereum's Blockchain*. 4 December 2017. August 2018. <https://www.coindesk.com/loveable-digital-kittens-clogging-ethereums-blockchain/>.

Hoffman, Chris. *Why It's Nearly Impossible to Make Money Mining Bitcoin*. 17 April 2018. 6 August 2018. <https://www.howtogeek.com/349033/why-it%E2%80%99s-nearly-impossible-to-make-money-mining-bitcoin/early-impossible-to-make-money-mining-bitcoin/>.

Infante, Roberto. *Building Ethereum ĐApps*. Manning Publications, 2018.

instructables.com. *Understanding How ECDSA Protects Your Data*. 28 December 2014. 27 July 2018. <https://www.instructables.com/id/Understanding-how-ECDSA-protects-your-data/>.

Intel SGX. n.d. August 2018. <https://software.intel.com/en-us/sgx/details>.

ipfs.io. n.d. 21 August 2018.

Jentzsch, Christoph. *The History of the DAO and Lessons Learned*. 24 August 2016. August 2018. <https://blog.slock.it/the-history-of-the-dao-and-lessons-learned-d06740f8cfa5>.

Lehikoinen, Tomi. *This summer, fishing in Finland means food traceability on the menu*. 2 July 2018. August 2018. <https://www.ibm.com/blogs/blockchain/2018/07/this-summer-fishing-in-finland-means-food-traceability-on-the-menu/>.

Martin, Keith M. *Everyday Cryptography: Fundamental Principles and Applications. 1st Ed*. Oxford: Oxford University Press, 2012.

Metz, Cade. *Google DeepMind's Untrendy Play to Make the Blockchain Actually Useful*. 3 November 2017. September 2018. <https://www.wired.com/2017/03/google-deepminds-untrendy-blockchain-play-make-actually-useful/>.

Middleton, Dan. *What's a Transaction Family?!* 22 June 2017. August 2018. <https://www.hyperledger.org/blog/2017/06/22/whats-a-transaction-family>.

Moos, Mitchell. *T-Mobile Unveils the Next Identity Platform, a Blockchain-Powered Database*. 28 March 2018. August 2018. <https://cryptoslate.com/t-mobile-next-identity-platform/>.

oraclize.it. *http://docs.oraclize.it*. n.d. 21 August 2018.

Orcutt, Mike. *MIT Technology Review*. 25 Aprit 2018. 31 July 2018. <https://www.technologyreview.com/s/610836/how-secure-is-blockchain-really/>.

Rapier, Graham. *The SEC has shut down another ICO — this time an alleged $600 million scam in Texas* . 30 January 2018. August 2018. <https://nordic.businessinsider.com/sec-shuts-down-arise-bank-600-million-alleged-ico-scam-dallas-texas-2018-1?r=UK&IR=T>.

Ray, James. *Ethereum Introduction*. 4 August 2018. 8 August 2018. <https://github.com/ethereum/wiki/wiki/Ethereum-introduction>.

—. *Proof of Stake FAQs*. n.d. August 2018. <https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQs>.

Roberts, Jeff John. *Decentralized Apps Like CryptoKitties Are Stalling*. 21 August 2018. August 2018. <http://fortune.com/2018/08/21/dapps-cryptokitties-augur-bancor/>.

Rosenbaum, Kalle. *Grokking Bitcoin*. Manning Publications, 2018.

Russinovich, Mark. *Announcing the CoCo Framework for enterprise blockchain networks*. 10 August 2017. September 2018. <https://azure.microsoft.com/en-us/blog/announcing-microsoft-s-coco-framework-for-enterprise-blockchain-networks/>.

Sawtooth Docs. 2017. Intel Corporation. August 2018. <https://sawtooth.hyperledger.org/docs/core/releases/1.0.5/introduction.html>.

Securing your wallet. 2018. 22 July 2018. <https://bitcoin.org/en/secure-your-wallet>.

Solidity Docs. 2018. 16 August 2018.

swissinfo.ch. *Switzerland's first municipal blockchain vote hailed a success*. 2 July 2018. 23 August 2018. <https://www.swissinfo.ch/eng/business/crypto-valley-_-switzerland-s-first-municipal-blockchain-vote-hailed-a-success/44230928>.

tlsnotary.org. *https://tlsnotary.org/*. n.d. 21 August 2018.

truffleframework.com. *https://truffleframework.com/docs/truffle/overview*. n.d. 22 August 2018.

Trón, Viktor, et al. "swap, swear and swindle incentive system for swarm." *Ethersphere Orange Papers*. May 2016.

uport.me. n.d. August 2018. <https://www.uport.me/>.

Vagle, Jeffrey L. "A Gentle Introduction to Elliptic Curve Cryptography." November 2000. *University of Pennsylvania Law School.* July 2018. <https://www.law.upenn.edu/cf/faculty/jvagle/workingpapers/A%20Gentle%20Introduction%20to%20Elliptic%20Curve%20Cryptography.pdf>.

Waldman, Jonathan. "Blockchain - Blockchain Fundamentals." *MSDN Magazine* 2018.

Valentin Vallois, Fouad Amine Guenane. "Bitcoin Transaction: From the Creation to Validation, a Protocol Overview." *Cyber Security in Networking Conference (CSNet)*. Paris: IEEE, 2017. 7.

vic, Victor. "https://commons.wikimedia.org/wiki/File:Card_games_and_game_tokens_01.jpg." Wikimedia, 25 January 2010. Photo.

Wood, Gavin. "ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER." Yellow Paper. 2018.