A STUDY OF BITCOIN AND BLOCKCHAIN

by

Jingyi Cai

Bachelor of Mathematics, University of Waterloo, 2014

A thesis presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Applied Mathematics

Toronto, Ontario, Canada, 2019

A STUDY OF BITCOIN AND BLOCKCHAIN

Master of Science, 2019

Jingyi Cai

Applied Mathematics

Ryerson University

Abstract

Blockchain technology is a distributed database and a public ledger. It records every transaction that has been made from its inception. Once entered, these records cannot be modified or erased. The technology utilizes various algorithms of a cryptographic nature to reach a consensus. These cryptograhic functions also ensure the integrity and authenticity of data that has been interchanged across the network. Because of these features, blockchain technology has been implemented into various financial and non-financial fields.

In this thesis, we introduce the mathematical foundations of Bitcoin, and different cryptographic functions that are used in Blockchain, especially elliptic curve multiplication. We construct two MATLAB models to study the fork events which is the one of typical consensus problems in the system. Moreover, we use graph theory and MATLAB models to represent and describe the Bitcoin protocols.

Acknowledgements

I would first like to express my gratitude to my supervisor Dr. Anthony Bonato for his continuous support and feedback during my research, and for his patience, guidance, and advice throughout my Masters degree studies.

Additionally, I would like to thank Dr. Alexey Rubstov and Dr. Dejan Delić for agreeing to serve as my thesis committee. I would also like to thank the faculty and staff in the Department of Mathematics at Ryerson University for their support in the past two academic years.

Finally, a very special thank to my family. This thesis would not have been possible without their encouragement, trust and unconditional love over years.

Table of Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

1.1. Motivation

We live in an era where technology rapidly develops. E-commerce is replacing traditional retailing. Tesla has developed a full self-driving vehicle and surgical robots have been adopted in operations. It is not surprising that technology innovations are changing traditional lifestyles. In particular, financial technology, also known as *Fintech*, is revolutionizing the financial sector.

In October 2008, Satoshi Nakamoto published the paper “Bitcoin: A peer-to-peer electronic cash system” [**47**]. In this paper, he came up with a new electronic payment system, which reduces a third party's involvements. This was the first time *Bitcoin* was introduced to the world. In 2009, one US dollar Bitcoin exchange rate was $1,309.03$ bitcoins [**42**]. No one knew that then this new currency would become so popular. At the time of writing this thesis, one Bitcoin was worth more than $\$12,000$ US dollars [**7**].

With the growing interest in Bitcoin, *blockchain*, the technology underlying Bitcoin, has become increasingly popular. In 2015, IBM announced that it will join an Open Ledger Project to improve business transactions by using blockchain technology [**39**]. Microsoft employed blockchain technology in its cloud computing service center Azure [**34**]. More recently, PricewaterhouseCoopers (also known as PwC) and Deloitte offered blockchain audit services [**19**, **20**].

Blockchain is a transparent and distributed shared system. It allows people to carry out transactions across the network while decreasing the risk of fraud. Cryptographic technology underpins blockchain.

1.2. Bitcoin Network

1.2.1. What is Bitcoin? *Bitcoin* was proposed in 2008, and was based on the idea of allowing individuals to perform transactions with one another without having the interference of a third party [**47**]. In January 2009, Satoshi Nakamoto received a reward of fifty bitcoins for mining the first blockchain block, *block* 0, also called the *genesis block* [**42**]. This has set up the foundation of the Bitcoin network.

Bitcoin uses several different kinds of novel technology. First, Bitcoin is a peer-to-peer (P2P) and a distributed ecosystem. In particular, every user or node equally serves the network. Transactions are received and propagated by users instead of by clearinghouses [**47**]. Also, Bitcoin is not regulated by any central authority such as a government or bank, which reinforces the P2P architecture. Second, Bitcoin is anonymous and the system is based on cryptographic technology [**3**]. Cryptography is utilized in creating accounts, protecting users and finding new blocks.

Third, Bitcoin can be sold, purchased, or even exchanged with other currencies. The total units are fixed to 21 million bitcoins, with the reward decreasing every four years at the rate of 50% [**3**]. This has controlled the inflation efficiently. Bitcoin decentralization leads to a situation that transactions could happen in different nodes at the same time. The traditional paper-based ledgers are no longer fit for Bitcoin. It needs to employ a database that is accurately and consistently recorded across the entire network. Last but not least, Bitcoin uses blockchain technology as its digital ledger.

1.2.2. Wallet and Key pairs. As we mentioned in the previous section, Bitcoin is an electronic payment system. Payment processing plays a key role over the internet. *Bitcoin address*, *key pairs*, and *digital signatures* are the three indispensable elements to establish a valid transaction [**3**]. Each key pairs contains a *Private key* and a *Public key.* A public key is generated from the private key, and it is used in Bitcoin address generation through hash functions (defined below in Chapter 2). The order of generation is irreversible, and therefore, each private key has its unique public key and address. We can think of the public key as similar to a bank card, and the address as similar to the digits on the card. The private key can be considered as the signature on the back of the card which shows the ownership of the account. The *wallet* is the database to store the keys and addresses [**3**]. The last step to finalize the transaction is to validate it with digital signatures so that the transactions can be listed in the blockchain.

1.3. Graph Theory

Before we delve deeper into Bitcoin and blockchain, we need foundational concepts in graph theory. A *graph* $G = (V(G), E(G))$ consists of a nonempty set of nodes $V(G)$, also called the *vertex* set, and an *edge* set $E(G)$, where each edge is associated with a pair of vertices called its *endpoints*. An edge uv is said to be *incident* on its endpoints if vertices u and v form an edge. We can also say that u and v are *adjacent*. The figure below represents a graph G.

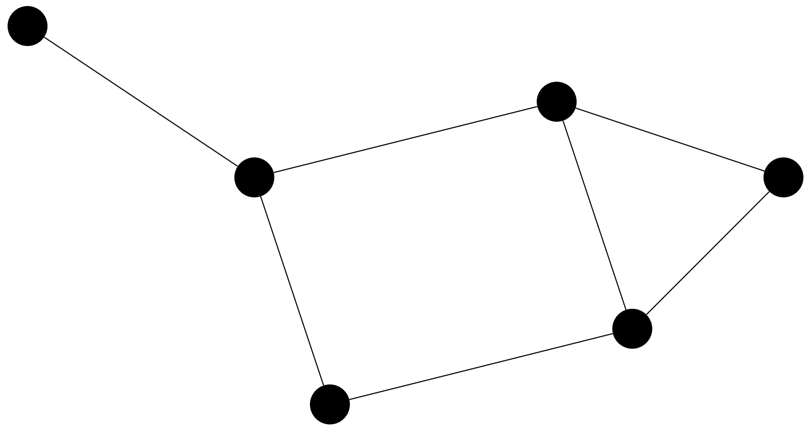


FIGURE 1.1. An example of a graph G.

Given a graph $G = (V(G), E(G))$, the number of vertices $|V(G)|$ is the *order* of the graph, and the number of edges $|E(G)|$ is the *size* of the graph. The graph G is finite when the order and the size are finite. We only consider finite graphs in this thesis. Further, we have that

$$|E(G)| \leq \binom{|V(G)|}{2}.$$

An edge with only one endpoint is called a *loop*, and a graph G without loops or multiple edges is called a *simple* graph. A *digraph* consists of a nonempty set of vertices, and a set of edges, where each edge has an orientation associated with it.

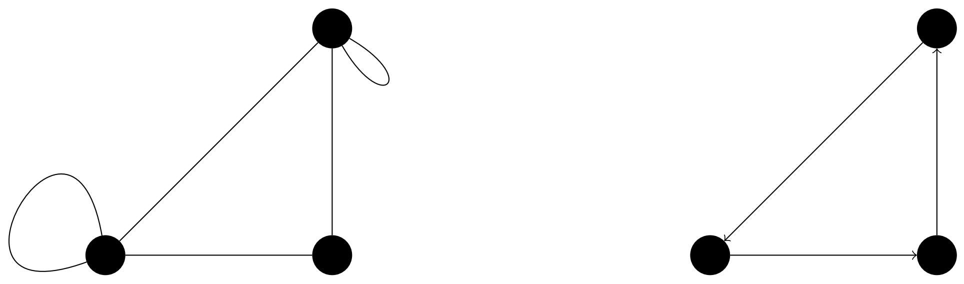


FIGURE 1.2. Examples of loops and a digraph.

Given a graph $G = (V(G), E(G))$ and $v \in V(G)$, the *neighbor* set of v, written $N(v)$, is the set of all the vertices adjacent to v. The *degree* of a vertex v, denoted by $deg(v)$, is the number of edges that incident on v. For instance, the $deg(v)$ in Figure 1.3 is 2. The *total degree* of a graph G is the sum of the degrees of vertices in $V(G)$.

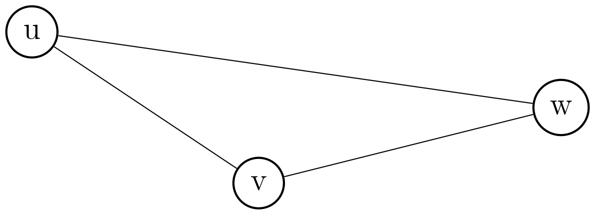


FIGURE 1.3. An example of a graph.

The degree of a vertex is one of the important parameters in graph theory. *The First Theorem of Graph Theory*, also known as *Handshaking Theorem*, states the mathematical relationship between the degrees and the size of a graph. The proof of the theorem is a part of folklore, and is included for completeness.

THEOREM 1.3.1. *(First Theorem of Graph Theory)*
For a graph $G = (V(G), E(G))$

$$\sum_{v \in V(G)} \deg(v) = 2|E(G)|.$$

PROOF. Each edge is associated with a pair of vertices. When we sum all the degrees, we would need to count each edge twice. □

The following corollary, also part of folklore, follows from Theorem 1.1.

COROLLARY 1.3.2. *For any graph G, the number of vertices of odd degree is even.*

1.3.1. Special Graphs.

The *path* denoted by P_n consists of a sequence of n vertices $(v_1, v_2, ..., v_n)$, such that v_i is adjacent to v_{i+1} for $1 \leq i \leq n-1$. The order of P_n is n and its size is $n-1$. The *length* of a path is the number of its edges. The *distance* $\mathrm{d}(u, v)$ between u and v is the length of the shortest path in a graph G.

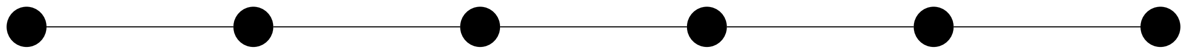


FIGURE 1.4. An example of a path P_6.

A graph G is called *connected* if there exists a path between any two distinct vertices. A *cycle* is a closed path with order $n \geq 3$, denoted by C_n. In a cycle C_n, every vertex has degree 2.

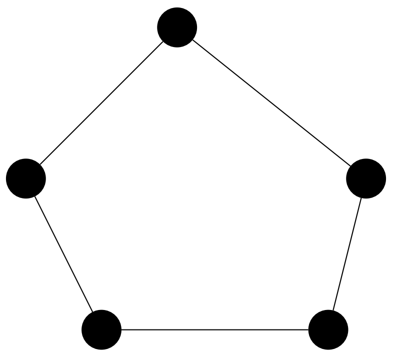


FIGURE 1.5. Cycle C_5.

A graph G is a *clique* if every two vertices are adjacent with each other. A clique of order n is denoted by K_n. Cliques are sometimes called the *complete* graphs. The size of K_n is $\binom{n}{2} = \dfrac{n(n-1)}{2}$.

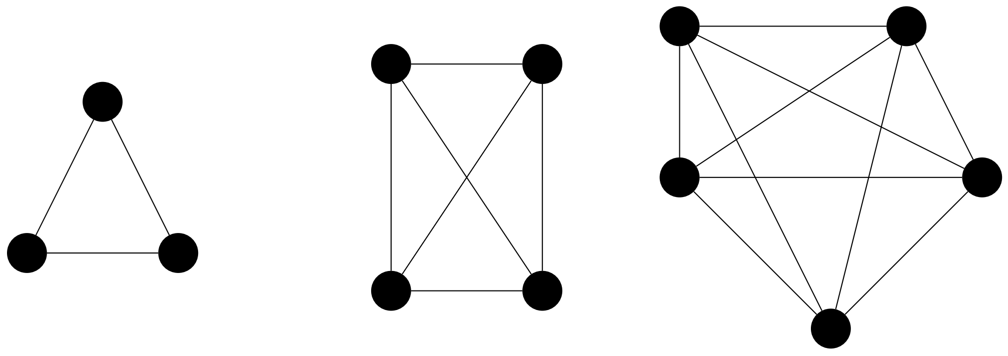


FIGURE 1.6. The graphs K_n for $n = 3, 4, 5$.

A graph G is a *tree* if it is a connected, acyclic graph. A tree of order n has $n - 1$ edges.

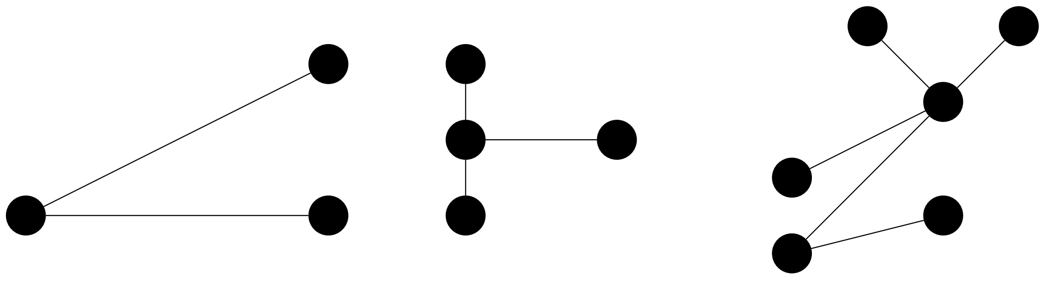


FIGURE 1.7. Trees with $n = 3, 4, 6$.

A graph can also be used to visualize the *complex networks.* The *web graph* is an example of a complex network, where vertices represent web pages, and edges represent the link between the pages. We may also consider the Twitter network as an example of a complex network, where nodes represent users, while edges represent followers. A complex network satisfies the following four properties [**11**]:

1) *Large-scale.* If users represent nodes, and the connections between users represent edges, then the order and the size of the graph are

extremely large. For example and according to [**29**], Facebook had 2.32 billion monthly active users in the fourth quarter of 2018.

2) *Evolving over time.* Unlike traditional graphs, where the vertex set and the edge set are fixed, real-world graphs are often dynamic. The vertices and edges are changing over time in complex networks.
3) *Small world property.* This was first introduced by Watts and Strogatz in 1998 [**54**]. The small world property states that a graph of order n demands a low diameter of $O(\log n)$ and a higher clustering coefficient compared with a random binomial graph of the same order and average degree.
4) The last property of a complex network are *power-law degree distributions.* Given a graph $G = (V(G), E(G))$, a non-negative integer k, the number of vertices of degree k in G is denoted by N_k. A power-law degree distribution means that N_k is proportional to k^{-b} for a fixed constant $b > 2$. Informally, a power law degree distribution implies that most vertices in a complex network have low degree, but some have very high degree. Therefore, the distributions also exhibit heavy tails.

For more background on graph theory and complex networks, the reader is directed to [**11**, **57**].

1.4. Probability Theory

We next introduce elementary concepts in probability theory. A *probability space* $\mathcal{S} = (\Omega, \mathcal{F}, P)$ consists of three elements. The *sample space* is a nonempty set of all possible outcomes of an experiment, denoted by Ω. There is also the set of random events $\mathcal{F}$, and the *probability function* $\mathbb{P}$. The probability function measures the likelihood that an event will occur, and we define $\mathbb{P} : \mathcal{F} \longrightarrow \mathbb{R}$, which satisfies the following properties.

(1) For all events $E, 0 \leq \mathbb{P}(E) \leq 1$.

(2) $\mathbb{P}(\Omega) = 1$.

(3) If $E_1, E_2, E_3, ...$ are disjoint or mutually exclusive events, such that $E_i \cap E_j = \emptyset$, then $\mathbb{P}(\bigcup_{i=1}^{\infty} E_i) = \sum_{i=1}^{\infty} \mathbb{P}(E_i)$.

For the proof of the following lemma, the reader is directed to [**9**].

Lemma 1.4.1. *Let $\mathcal{S}(\Omega, \mathcal{F}, P)$ be a probability space, E_1,$E_2 \in \mathcal{F}$,then*

(1) If $E_1 \subseteq E_2$, then $\mathbb{P}(E_1) \leq \mathbb{P}(E_2)$.

(2) For an empty set $\emptyset$, $\mathbb{P}(\emptyset) = 0$.

(3) $\mathbb{P}(E_1) = 1 - \mathbb{P}(\Omega \setminus E_1)$.

(4) If E_1, E_2 are disjoint, such that $E_1 \cap E_2 = \emptyset$, then $\mathbb{P}(E_1 \cup E_2) = \mathbb{P}(E_1) + \mathbb{P}(E_2)$.

(5) $\mathbb{P}(E_1 \cup E_2) = \mathbb{P}(E_1) + \mathbb{P}(E_2) - \mathbb{P}(E_1 \cap E_2)$.

Events E_1, E_2 are *independent* if $\mathbb{P}(E_1 \cap E_2) = \mathbb{P}(E_1)\mathbb{P}(E_2)$ holds, which means that the probability of one event happens does not affect the probability that the other occurs. For a sequence of independent events $E_1, E_2, \dots E_m \dots$, we have that

(1) $\mathbb{P}(\bigcap_{i=1}^{m} E_i) = \prod_{n=1}^{m} \mathbb{P}(E_i)$.
(2) $\mathbb{P}(\bigcap_{i=1}^{\infty} E_i) = \prod_{n=1}^{\infty} \mathbb{P}(E_i)$.

A *random variable* $\mathcal{X}$ on a probability space $(\Omega, \mathcal{F}, P)$ is a function $\mathcal{X} : \Omega \longrightarrow \mathbb{R}$. There are two types of random variables, discrete and continuous. A discrete random variable is a random variable whose set of values could come from the entire $\mathbb{N}$. For example, suppose we toss a coin. Let $\mathcal{X}$ represents the number of experiments to see the first head. We only consider discrete random variables in this thesis, unless otherwise specified.

The *expectation value* of a random variable X on a finite probability space $(\Omega, \mathcal{F}, P)$ is defined as

$$\mathbb{E}(X) = \textstyle\sum_{x \in \Omega} x \ \mathbb{P}(x).$$

For $i = 1, 2, \dots n$, let a_i be real number, and let X_i be a random variable. We then have that the following property holds

$$\mathbb{E}(\sum_{i=1}^{n} a_i X_i) = \sum_{i=1}^{n} a_i \ \mathbb{E}(X_i).$$

For more background of probability theory, see [**9**, **21**].

1.5. Summary of Thesis

This thesis is composed of five chapters. In Chapter 1, we presented an introduction to Bitcoin and blockchain, as well as the basic terminologies of graph theory and probability theory. In Chapter 2, we will consider elliptic curve cryptography and how it generates key pairs. We will also explain the transaction constructions in Bitcoin. Chapter 3 focuses on blockchain, in particular, the block aggregation and the proof-of-work algorithm. Also, we will introduce two new models to describe fork events. Chapter 4 describes Bitcoin protocols and new simulations of the protocol graphs. There will also be examples of blockchain applications. In the last chapter, we will summarize our results and present open problems from the thesis.

CHAPTER 2

Cryptography

2.1. Introduction

Cryptology is the study of secure communication, and the field utilizes techniques from mathematics, computer science, and information theory. Cryptology is comprised of *cryptography* and *cryptanalysis* [**41**]. Cryptography focuses on message encryption, where a message can only be seen by the designated receiver. Cryptanalysis focuses on breaching the cryptographic system and falsifying messages [**41**].

A cryptographic system consists of five elements [**33**]:

1) *The plain text*, which is the original message that the sender wants to send.
2) The *cipher text* is the encrypted message.
3) The method to convert plain text into ciphertext is called *encryption.*
4) The method to convert ciphertext into plain text is called *decryption.*
5) The last is the *cryptographic key.*

Before the 1970s, senders and receivers were all using the same type of cryptographic keys during the encryption and decryption, called the symmetric key system [**13**]. Exchanging a key safely was the crucial requirement in the symmetric key system. However, as the number of users increased, it also increased the difficulty of key management. In 1976, Diffie and Hellman proposed an idea of an *asymmetric key system* in their paper "New Directions in Cryptography" [**24**].

2.2. Public Key Cryptography

Before we go into details on the asymmetric key system, we need to define the concepts used in the system. A *one-way function* is a function that can be feasibly computed on every input, but computationally infeasible to find its inverse. Prime factorization is an example of a one-way function. Let X and Y be non-empty sets, with X representing the set of *inputs* and Y representing the set of *outputs*, a function $f : X \longrightarrow Y$ is called a *trap-door one-way function*, if it satisfies the following properties.

(1) For a given input $x \in X$, we can compute the corresponding $f(x)$.
(2) For a given image $y \in Y$, it is computationally infeasible to find the corresponding x such that $y = f(x)$.
(3) Given the *trap-door information* δ and an image $y = f(x)$, we can find the corresponding x feasibly.

Prime factorization is also an example of a trap-door one-way function. Finding the prime factors of a sufficiently large integer is difficult, but if we know one or more of these factors, the other factors can be computed easily. The trap-door information could be a number, or it could also be the random bit string used to produce the key pairs [**24**].

An asymmetric key system requires two different pairs of keys during the encryption and decryption procedures, which increases the security capability and overcomes the weakness of the symmetric key system. Each pair of keys contains a *public key* that can be shared to others and a *private key* which is only known by the owner. An individual A who wants to send a message to an individual B needs to use B's public key to encrypt the message. The latter can use the corresponding private key to decrypt the message. In [**24**], Diffie and Hellman proposed to use a trap-door one-way function to generate such key pairs, where private key as input and the public key is the output of the trap-door one-way function. Since it is computationally infeasible to solve the private key given a known public key, the public keys are no longer kept in secret. Therefore, this system is also referred to the *public key cryptography.* This thesis will focus only on public key cryptography used in Bitcoin. For more background on public key cryptography, the reader is directed to [**12, 13, 24, 33, 41**].

2.2.1. Public Key Cryptography In Bitcoin. *Elliptic curve cryptography* (ECC) over a finite field is one of the approaches to public key cryptography. It based on the discrete logarithm problem and has been applied widely [**15**, **35**]. In order to understand the procedures of this approach, we first need several definitions.

2.2.2. Terminology. A *hash function* is any function that takes an arbitrary number of bits data as input, and outputs the data with a fixed number of bits. An example of a hash function is the modulo function. Consider a set of integers $I = \{13, 19, 255, 8650\}$ with different digits, and a function

$$f(a) = a \pmod{200}$$

for every integer $a \in I$. Under the modulo operation, the function f will output a new set of 2-digit integers $\{13, 19, 55, 50\}$.

A *cryptographic hash function* is a one-way hash function. *Secure Hash Algorithm* (SHA) and *RIPE Message Digest* (RIPEMD) are two families of cryptographic hash functions that are implemented in Bitcoin [**3**].

Let S be a non-empty set, a *binary operation* $*$ on the set S is a mapping:

$$* : S \times S \to S.$$

The operation $*$ may satisfy the following properties for all $a, b, c \in S$:

1) (Commutativity). $a * b = b * a$.
2) (Associativity). $(a * b) * c = a * (b * c)$.

For instance, addition and multiplication are two binary operations on the sets $\mathbb{R}$. For any elements $a, b, c \in \mathbb{R}$, these operations are satisfy commutative

$$a + b = b + a, a \cdot b = b \cdot a,$$

and associative

$$a + (b + c) = (a + b) + c, a \cdot (b \cdot c) = (a \cdot b) \cdot c.$$

An *abelian group* is a non-empty set G with a binary operation $*$ on G, such that

1) $*$ is commutative and associative.
2) (Identity) For each $a \in G$, there exists an identity element $e \in G$ such that, $a * e = e * a = a$ holds.
3) (Inverse) For each $a \in G$, there exists an inverse element $i \in G$ such that, $a * i = i * a = e$ holds, where e is the identity element.

For example, the integers under ordinary addition from an abelian group. The negative integers are the inverse elements, and 0 is the identity element. The non-zero real numbers under ordinary multiplication from a group as well. For every non-zero real number, its inverse is this approach inverse element, and 1 is the identity element.

A *field* is a non-empty set F with two binary operations addition $(+)$ and multiplication $(\cdot)$ satisfies the following properties.

(1) The operation $+$ forms an abelian group.
(2) For all elements $x, y \in F$, $x \cdot y = y \cdot x$.
(3) For all elements $x, y, z \in F$, $(x \cdot y) \cdot z = x \cdot (y \cdot z)$.

(4) There exists an identity element $1 \in F$ under multiplication, such that $1 \cdot x = x \cdot 1 = x$.

(5) For every $x \in F$ and $x \neq 0$, there is an element $x^{-1} \in F$ such that $x \cdot x^{-1} = 1$.

(6) (Distributivity) $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$.

For example, the set $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ of integers $\pmod 5$ is a field. Addition and multiplication of any two elements in the set $\mathbb{Z}_5 = \{0, 1, 2, 3, 4\}$ of integers $\pmod 5$ are given by the following tables.

TABLE 2.1. Addition in $\mathbb{Z}_5$.

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1

TABLE 2.2. Multiplication in $\mathbb{Z}_5$.

$\times$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3

Furthermore, the sum and the product of all elements of a set are $1 + 2 + 3 + 4 + 5 = 0 \pmod 5$, and $1 \times 2 \times 3 \times 4 = 4 \pmod 5$. The corresponding additive and multiplicative inverse elements of $\{1, 2, 3, 4\}$ are $\{4, 3, 2, 1\}$, and $\{1, 3, 2, 4\}$. We note that for any prime number p, the set $\mathbb{Z}_p$ of integers $\pmod p$ is always a field.

A *finite field* is a field F with a finite number of elements. The cardinality of a field F is called the *order* of F. Furthermore, the order of a finite field F is always a prime number p or a power of prime p^n, where n is a positive integer, and written as F_p. For more background of abstract algebra, see [**26**, **38**, **46**].

2.2.3. Elliptic Curve Multiplication.

An *elliptic curve* E is a curve defined on the plane by an equation of the form

$$E : y^2 = x^3 + ax + b.$$

where a and b are real numbers, such that $4a^3 + 27b^2 \neq 0$. The *points at infinity* are denoted as ∞ or O. The figure below shows a graph of an elliptic curve over real numbers $\mathbb{R}$. It is generated by the online graphing tool Desmos [**22**].

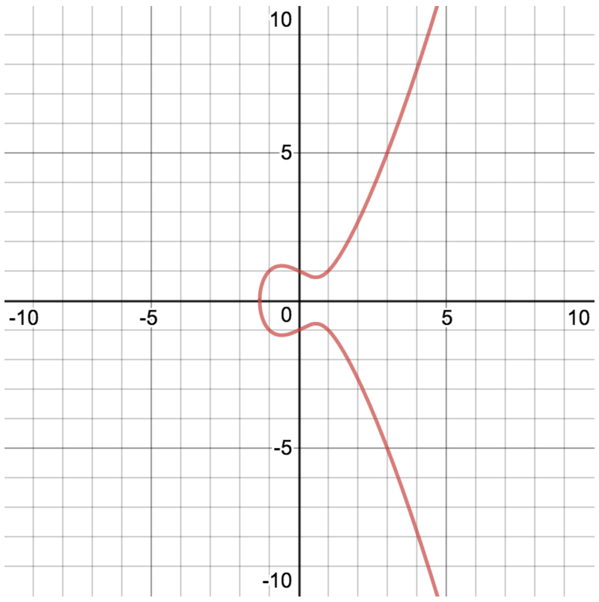


FIGURE 2.1. An example of an elliptic curve $y^2 = x^3 - x + 1$.

Elliptic curve multiplication (ECM) is kind of multiplication derived from an elliptic curve. It is accomplished by adding a point to itself along an elliptic curve repeatedly. Let E be an elliptic curve defined over finite field. Suppose we are given two points P and Q on an elliptic curve, and a positive integer n. Adding a point P to itself repeatedly can be written as

$$Q = np = \underbrace{P + P + P + \cdots + P}_{\text{adding P to itself n times}}$$

. Given P and Q on the curve, can we fine an integer n such that $Q = nP$. This problem is also known as the *discrete logarithm problem* (DLP) on an elliptic curve (ECDLP). There is no feasible algorithm to compute ECDLP [**32**]. Elliptic curve multiplication over a finite field is computationally expensive, Bitcoin uses it as a means of one-way function to generate key pairs and addresses. To better interpret ECM in Bitcoin, we first consider operations on an arbitrary elliptic curve over $\mathbb{R}$.

We first define an abelian group over elliptic curve E with addition operation such that

1) The points on an elliptic curve E are the elements of the group.
2) For any two points P_1, P_2, we have $P_3 = P_1 + P_2$ also on an elliptic curve E.
3) The point at infinity O is an identity element, such that for any point P, $P + O = P$.

4) The inverse element of point P is the point reflected in the x-axis, denoted as $-P$.

The following figures are geometric representations of addition operations on an elliptic curve, they are generated by an online elliptic curve calculator tool from Desmos [**28**].

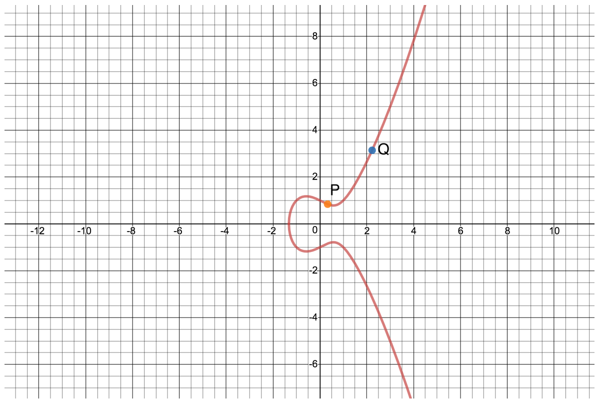


FIGURE 2.2. Points P and Q are two distinct points on $E : y^2 = x^3 - x + 1$.

Given two points P and Q, we can find a line passing through P and Q. This line will intersect a point on an elliptic curve, denoted as R.

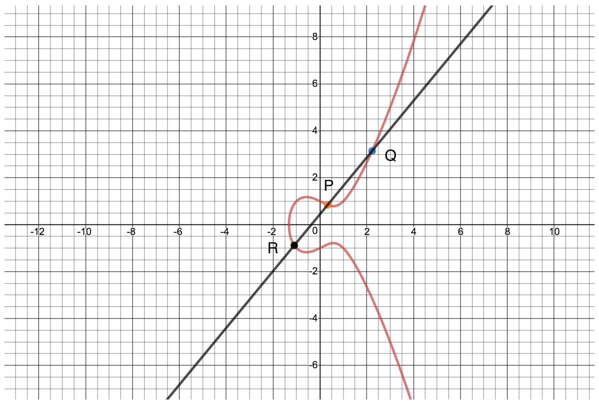


FIGURE 2.3. Addition of two distinct points P and Q.

Since it can be shown that an elliptic curve multiplication is a group (see [**15**]), by the associative property, we have that $(P+Q)+R = P+(R+Q) = O$, for any points P, Q, R on an elliptic curve. We may also write $P+Q = -R$.

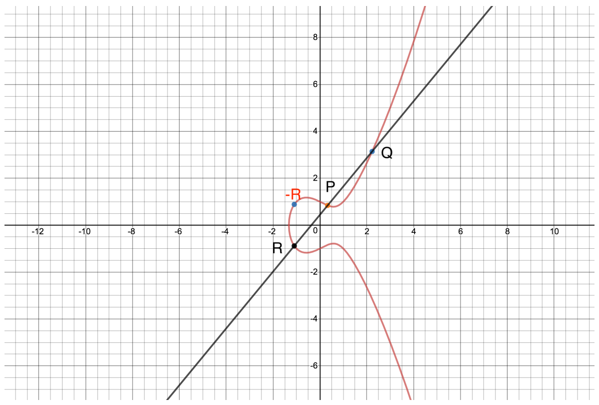


FIGURE 2.4. Point $-R$ is the resulting point.

We defined above the addition of two points, but we want to add a point to itself repeatedly instead of using distinct points. If points P and Q are the same points, then the line between the points will be considered as the tangent line at point P. Figure 2.5 is an example of addition of the same point on an elliptic curve.

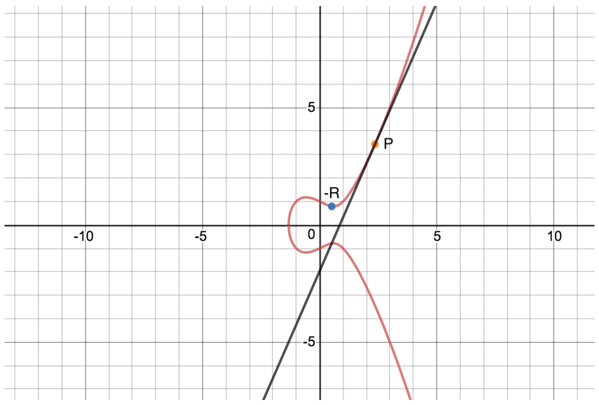
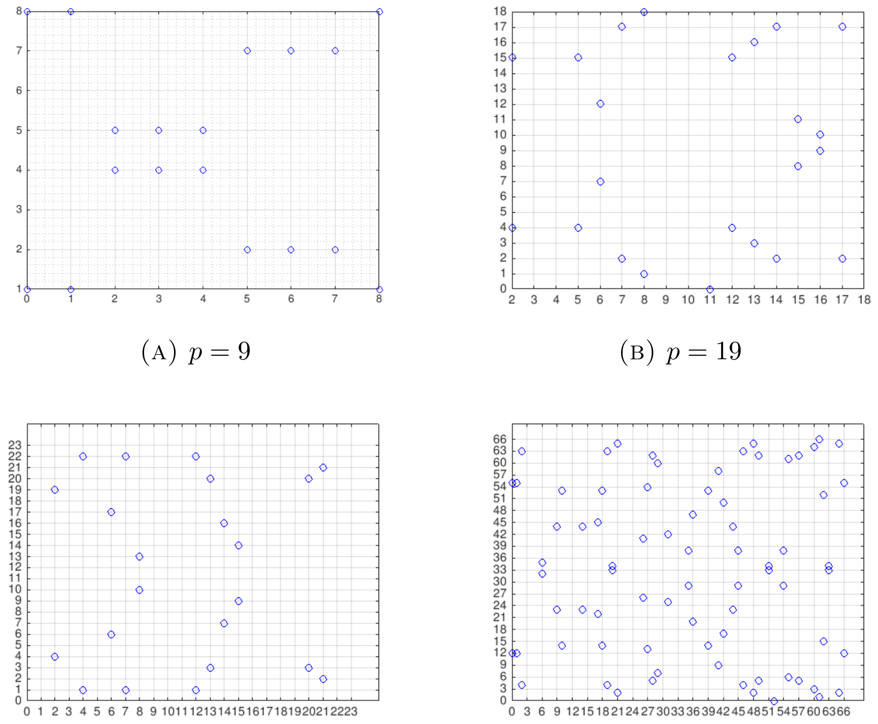


FIGURE 2.5. Addition of same point on $E : y^2 = x^3 - x + 1$.

Now, let us consider operations on an elliptic curve over finite field F_p with prime order p instead of $\mathbb{R}$. Since F_p is a non-empty set of elements, the graph of an elliptic curve over F_p is a discrete plot. The following figure is a Matlab [**43**] example of an elliptic curve $E : y^2 = x^3 - x + 10$ over F_p.



(A) $p = 9$ (B) $p = 19$

(C) $p = 23$ (D) $p = 67$

FIGURE 2.6. An example of an elliptic curve over F_p with $p = 9, 19, 23, 67$.

The calculations on the finite field are analogous to those with real numbers [**3**]. For more background on elliptic curve cryptography and the discrete logarithm problem, the reader is directed to [**3**,**15**,**35**,**50**,**53**].

2.2.4. Bitcoin Keys Generation. As described in Chapter 1, Bitcoin is a decentralized electronic cash system. In order to achieve the consensus in Bitcoin for all participants, it uses a specific elliptic curve $E : y^2 = x^3 + 7$, defined as *secp256k1* over F_p [**52**] with

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1.$$

With the large size of p, the graph of an elliptic curve secp256k1 is too complex and large to visualize.

As we mentioned in Chapter 1, the address is generated from the public key, while the public key is generated from the private key. The private key is a 256 bit random number generated by an underlying number generator system in Bitcoin. The public key is computed from a private key by using an elliptic curve multiplication. Let k_{priv} represent the private key, and let K_{pub} be the public key. The process can be expressed as

$$K_{pub} = G * k_{priv},$$

where G is a constant point on secp256k1, called the *generator point*. The equation is identical to the one in ECDLP. It is *irreversible*, which means by knowing the public key and the generator point, it is hard to find the corresponding private key. The bitcoin address is a string of alphanumeric characters, and it is converted from the public key K_{pub} by hashing and encoding. Bitcoin uses two different cryptographic hash functions *SHA256* and *RIPEMD160* to hash the public key. ECM and double hashing algorithms can be operated efficiently with a known private key. Figure 2.7 is adapted from [**3**], and it describes the generating procedure from public key to an address.

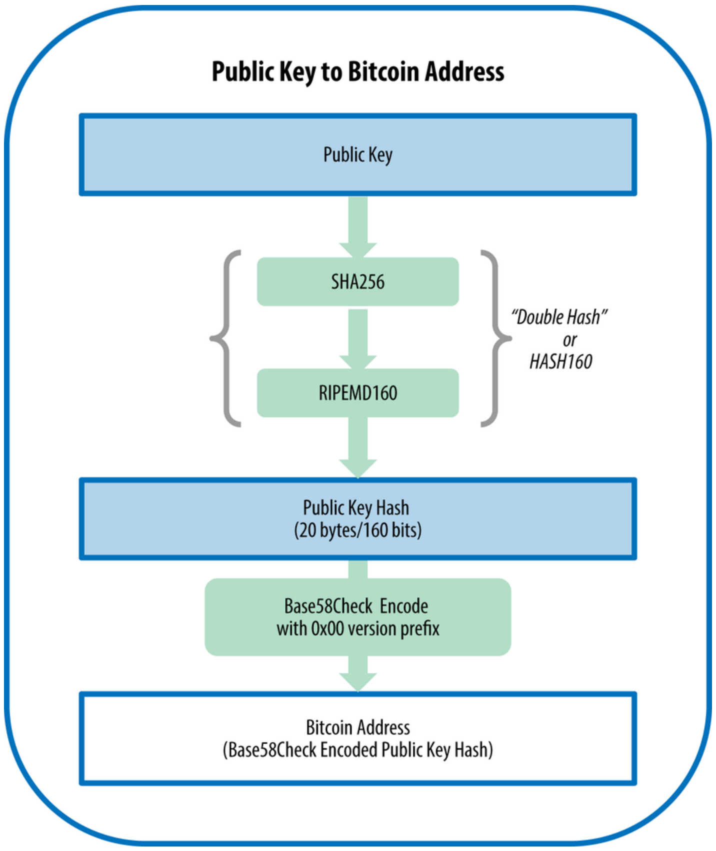

Public Key to Bitcoin Address
Public Key
SHA256
RIPEMD160
"Double Hash" or HASH160
Public Key Hash (20 bytes/160 bits)
Base58Check Encode with 0x00 version prefix
Bitcoin Address (Base58Check Encoded Public Key Hash)

FIGURE 2.7. The generating procedure.

2.3. Digital Signature

A *digital signature* is another approach to public key cryptography. It aims at binding the sender to the plain text together, like a handwritten signature, and it is an acknowledgement that the plain texts are authentic and integrated. A digital signature algorithm is designed to be signed by the sender and to verify by the receiver simultaneously, a digital signature is a number attached to the plain text.

The process consists of two parts. First, senders use their own private key to sign the plain text. Second, the encrypted signature-related text can only be decrypted by senders' public key. The private key is only known by its owner, so it is nearly impossible to imitate a signature but it is easy to verify it. For more information about digital signatures, see [**13**].

2.4. Transaction Construction

We now have all the necessary elements to construct a Bitcoin transaction. Bitcoin uses the *scripting language* for transactions [**51**]. A *script* is a list of conditions or constraints, and there are two types of scripts: the *locking script* and the *unlocking script.* A locking script is a list of conditions need to be met to spend the bitcoins, locking script locks the output to the destination address, while the unlocking script is the script satisfies the locking script conditions [**3**]. A Bitcoin transaction is a form of data transference, so it needs inputs and produces outputs. To operate a new Bitcoin transaction, the wallet will start to find the previous *unspent transaction outputs* (UTXO) and the corresponding unlocking script as inputs. An unspent transaction output is a form of currency in bitcoin; it is indivisible and is locked to its owner. The transaction will produce a new UTXO with its locking script as output.

For example, suppose you have a ten bitcoins UTXO and wish to spend five bitcoins. The only way to do it is to consume this ten bitcoins UTXO as an input, and the transaction produces two UTXOs with locking scripts as outputs: the first five bitcoins UTXO goes to payee's wallet, the second five bitcoins as a new UTXO back to your addresses. To aggregate the transactions into blockchain in a timely manner, most transactions include transaction fees. Transaction fees are the differences between the inputs and outputs. In this case, after consuming an entire ten bitcoin UTXO, it will produce three outputs: the first one goes to payee's wallet, second one back to our wallet as a new UTXO, and the last one with transaction fees goes to miner's wallet.

We next consider an example based on the *pay-to-public-key-hash* script to better explain the transaction constructions. The pay-to-public-key-hash (P2PKH) script is one of the common transaction scripts in Bitcoin [**3**]; the unlocking script of P2PKH is the owner's digital signature and the public key. Suppose Bob and Alice are two participants in Bitcoin. Alice wishes to send forty bitcoins to Bob in exchange for goods, and she has an exact amount of UTXO she needs to pay, the forty bitcoins UTXO in her wallet. To set up a transaction input, we need this forty bitcoins UTXO, her digital signature and public key, as shown in Figure 2.8.

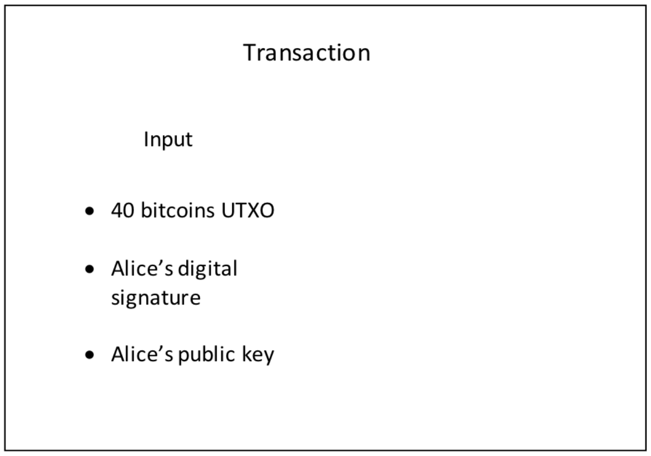


FIGURE 2.8. An example of a transaction input.

Alice pays forty bitcoins to Bob's address, the transaction will output a forty new bitcoins UTXO and a locking script, as shown in Figure 2.9.

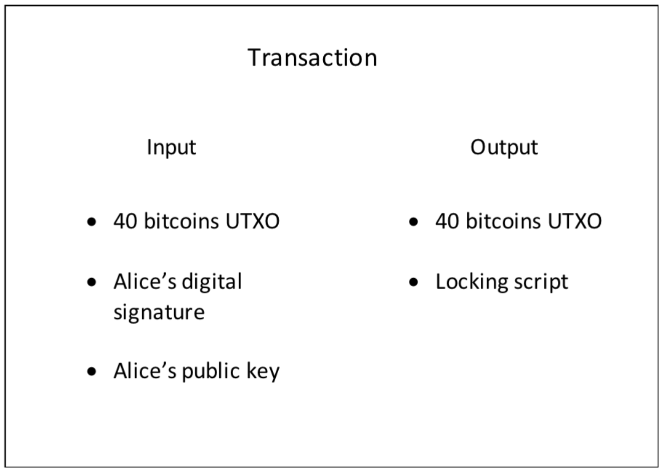


FIGURE 2.9. An example of a transaction.

Since UTXOs are indivisible, sometimes we may need to use multiple UTXOs as inputs, add them up to construct a transaction. Below is a transaction consumed by multiple UTXOs as inputs to cover the cost.

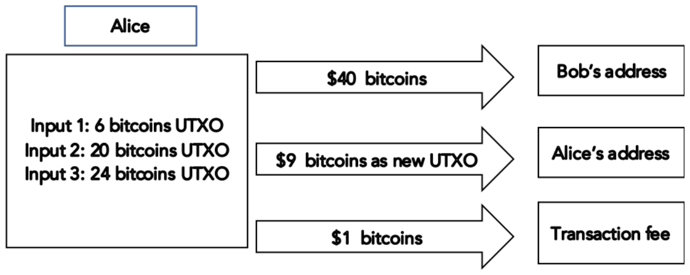


FIGURE 2.10. An example of a transaction with multiple inputs.

Alice will then broadcast this specific transaction into the Bitcoin network. Every node in the network starts to verify the transaction, and the verified transaction will be propagated to the next node. There is a list of requirements, which need to be checked before it is verified; see [**3**,**48**] for more details. The transaction will be propagated continuously until it reaches every node in the network, as shown in Figure 2.11. The verified transaction will be stored into a transaction pool, until now, this transaction has been verified but still unconfirmed. For more details on Bitcoin transactions, see [**3**].

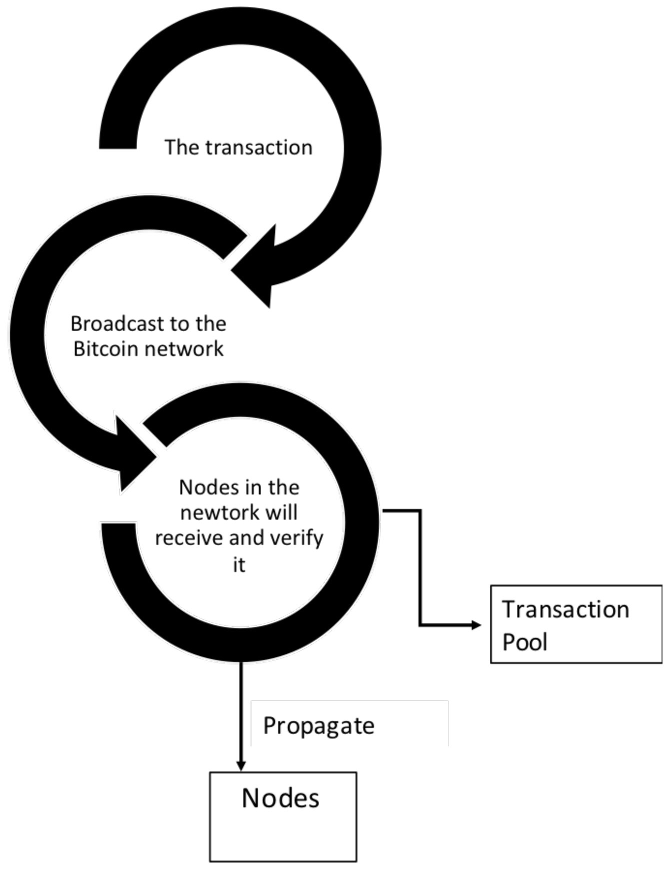

The transaction
Broadcast to the Bitcoin network
Nodes in the newtork will receive and verify it
Transaction Pool
Propagate
Nodes

FIGURE 2.11. Propagation.

CHAPTER 3

Blockchain

3.1. Introduction

With the growing impact of digitization on society, an increasing number of financial transactions are governed by mathematical algorithms. Blockchain was purposed as a tool to support the Bitcoin network in 2008 [**47**]. Blockchain enables participants to record, to track, and to verify transactions across the network independently. The blockchain data structure is a list of blocks of transactions [**3**]. More explicitly, it is a back-linked list data structure, so that every block is created on top of the existing chain. Every block is made of a unique block header and body, the block header summarizes the data in the block. The following figure is a geometric representation of blockchain structure.

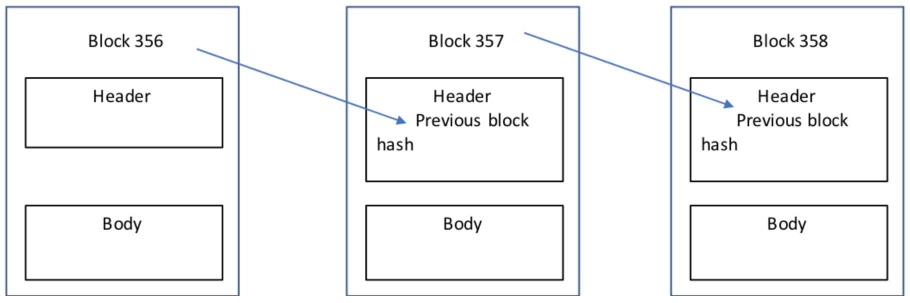


FIGURE 3.1. Blockchain structure.

The block header is composed of the following sets of metadata.

1) The hash of the previous block.
2) *The Merkle root*, which is the summary of transactions.
3) Difficulty target and *nonce* are the data used in the *proof-of-work* algorithm, which will be described in the next sections. A nonce is a variable with initial value zero.
4) A *timestamp*, which is the approximated creation time of the block.

The block header hash is the result of hashing the block header twice through the SHA256 algorithm, and it is a unique identifier of the block. Every block is linked to the previous one by including the hash of the previous block in the *block header*. Therefore, the chain of blocks can also be visualized as a sequence of hashes [**3**, **31**]. The block *height* is the second identity of the block, and it is the position of the block within the chain [**3**]. Since every block header contains a piece of hash information from the previous block, these blocks are connected consecutively through hash functions. It is nearly impossible to tamper with one block without affecting others. For more details in blockchain, see [**3**, **16**, **31**, **47**].

Blockchain is designed as a decentralized distributed ledger in Bitcoin. That is, every confirmed transaction is traceable and verifiable by each node within the network. However, the size of each block is limited.

In order to fulfill all requirements demanded by the system, blockchain employed the *Merkle tree* as its core data structure [**18**].

3.1.1. Merkle Tree. A *binary tree* is a tree where each node contains at most two children. The following figure is an example of a binary tree.

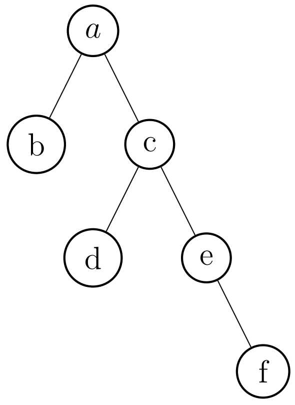


FIGURE 3.2. An example of a binary tree.

The *Merkle tree* is a hash-based binary tree data structure used for massive data storage and verification purposes. Every leaf node in a Merkle tree contains the hash of the data, and every non-leaf node is made by hashing its children nodes [**18**, **44**]. *The Merkle root* is the root of the Merkle tree. For example, given a set of transactions $\{A, B, C, D\}$, we may place this set into a Merkle tree structure through the hash function H, as follows.

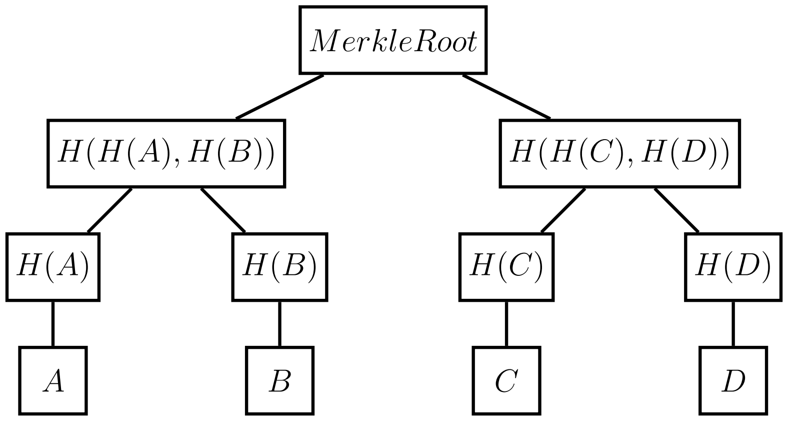


FIGURE 3.3. An example of a Merkle tree.

In Figure 3.3, the Merkle root represents a set of transactions with a single hash value. If we consider an enormous number of transactions followed by the same hashing algorithm, then the Merkle tree structure serves not only to manage the large data set efficiently but also to prevent data tampering. The other reason for employing this data structure is that it underpins the transaction validation processes.

For example, to verify a transaction D without looking at the full copy of transactions, participants first need to download the hash values $H(C)$ and $H(H(A), H(B))$, then compute the Merkle root to check if it same as the one recorded in the block.

For a block with n elements, the time to verify an arbitrary transaction is at most $2 \log 2n$ [**3**], Figure 3.4 is adapted from [**3**], and it describes a Merkle tree with 16 transactions. For more details on Merkle trees, the reader is directed to [**2, 3, 18, 37, 44, 45**].

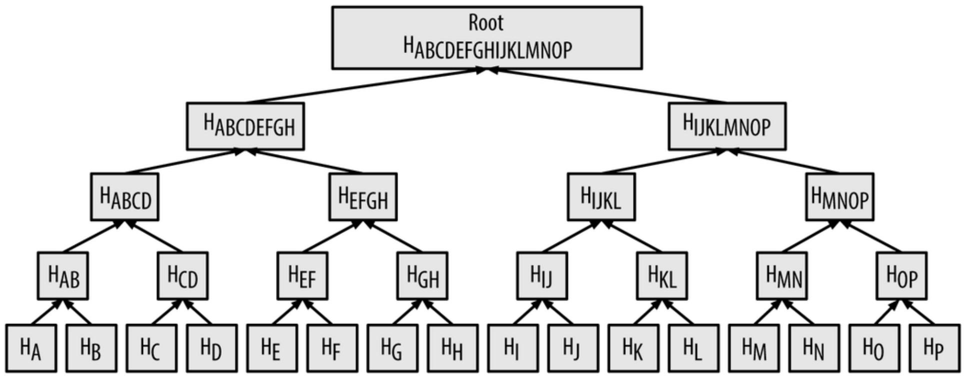


FIGURE 3.4. A Merkle tree with 16 transactions.

3.2. Block Aggregation

3.2.1. Transaction Aggregation. As we described in previous chapters, an unconfirmed transaction will be stored into a node's transaction pool and be transmitted to others. Nodes equally serve the network based on their functionality. A *mining node* is one of the nodes within the network which is responsible for aggregating transactions into blockchain. Mining nodes are rewarded for every new block is created, the process of creation is called *mining* [**3**, **47**]. Mining is not only the process of earning rewards, but also an approach to prevent fraudulent transactions. Figure 3.5 is a geometric representation of Bitcoin network, where edges represent the connection between the nodes.

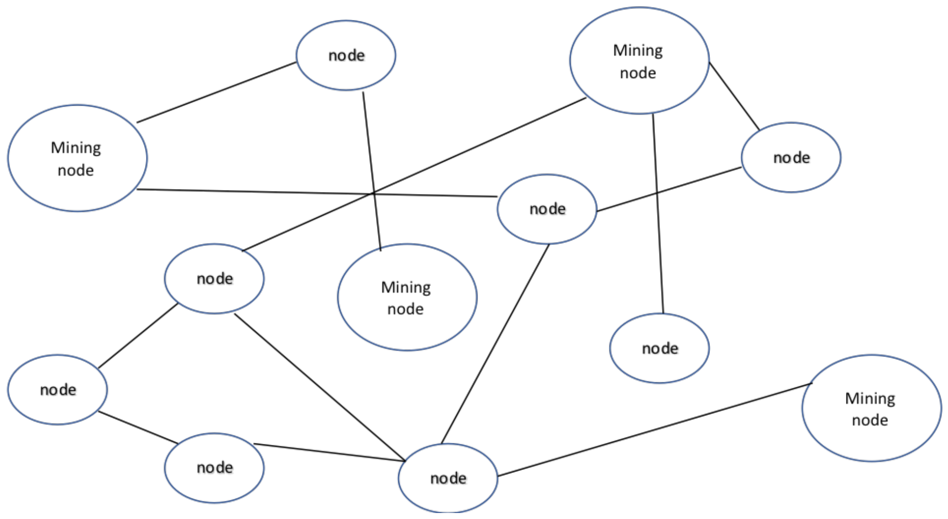
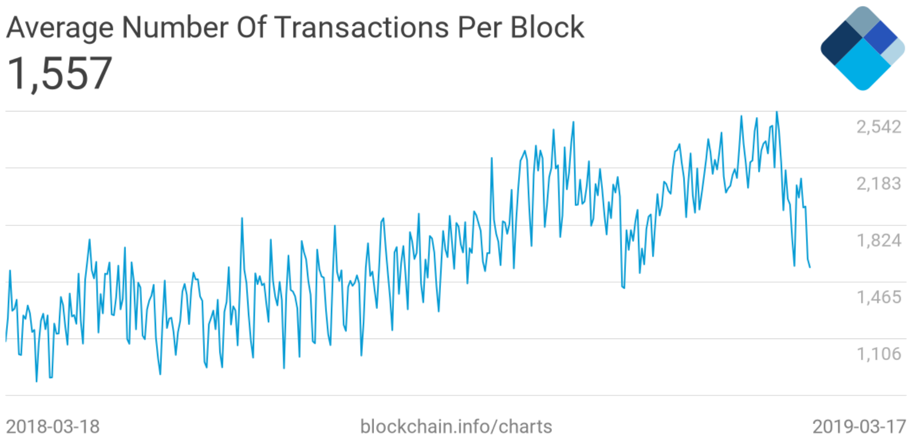
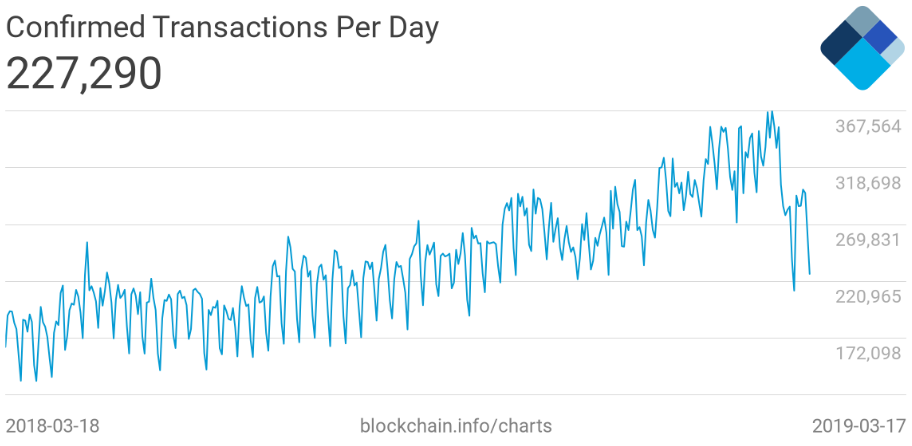


Figure 3.5. Geometric representation of a Bitcoin network.

Mining nodes receive, validate, and store a transaction like others. They will then start aggregating transactions and generating the *candidate block*, which is the block of the unconfirmed transactions from the transaction pool. Transactions are selected from the pool based on their priority as determined by the system. Once a corresponding Merkle root is computed, mining nodes need to complete the block header, as listed in the above section. Figure 3.6 is adapted from [**14**]. For more details in transactions aggregation, see [**3**, **47**]. In order to preserve the block's authenticity, the final step is to validate the completed candidate block by solving the *proof-of-work* algorithm.



(A)



(B)

FIGURE 3.6. Transactions summary from March 2018 to March 2019.

3.2.2. Proof-of-Work Algorithm. A *Proof-of-work*, or PoW algorithm was first introduced in [**27**]. The goal was to prevent junk emails by employing computationally expensive but efficiently verifiable functions or algorithms. More generally, the PoW algorithm increases the senders' costs, but it is relatively easier for recipients to verify [**40**].

The *Adam Back's Hashcash* [30, 47] is the foundation of the PoW algorithm used in Bitcoin mining process. Hashcash is a hash-based algorithm, and it was introduced by Adam Back in [4]. He suggested to add a variable called *nonce* into the original metadata as input, and produces a hash with some numbers of leading 0's [4, 30]. Bitcoin mining process implements a similar algorithm. That is, the algorithm will increase the nonce continuously, until it yields a hash value with required numbers of leading 0's [3, 47]. The algorithm hashes the candidate block header repeatedly with the incremental nonce, until the result fits the PoW criteria in which the hash value is less than the target one. PoW is the first consensus mechanism in Bitcoin [3, 30, 47].

The mining process can be also viewed as a number of random guesses. For example, let X be the random variable which represents a two-digit integer. The probability of guessing a two-digit number is $\frac{1}{90}$. Now let y be a two-digit integer where $y \geq 11$ represents the difficulty. The probability of $\mathbb{P}(X < y)$ equals to

$$\mathbb{P}(X = 10) + \mathbb{P}(X = 11) + \mathbb{P}(X = 12) + \cdots + \mathbb{P}(X = y - 1) = \frac{y - 10}{90}.$$

We can see that the probability of $\mathbb{P}(X < y)$ depends on y. That is, the time to solve the PoW depends on the Bitcoin *difficulty*. The difficulty is the measure of how difficult it is to find the right hash value [3]. It is determined by a 256-bit target value, and the equation of difficulty measure can be defined as [23]:

$$\text{Difficulty} = \frac{\text{difficulty\_1\_target}}{\text{current\_target}}.$$

where difficulty_1_target represents the pool difficult, and the current_target represents the current target value. Figure 3.7 shows the PoW algorithm. Since the desired hash value needs to meet the PoW criteria, the mining difficulty increases with a lower target value, and decreases with a higher one. In Bitcoin, the target value or the mining difficulty level is set to whatever a new block will be mined in ten minutes. Bitcoin has a global difficulty target, currently set at 6,068,891,541,676.553 [8]. Figure 3.8 is adapted from [14], it captures the changes of Bitcoin mining difficulty over a year. The difficulty is adjusted based on how much effort has been contributed by the mining nodes. As the network is growing, the number of mining nodes increase, and it becomes harder to solve the PoW. For more details in PoW algorithm, see [3, 30, 47].

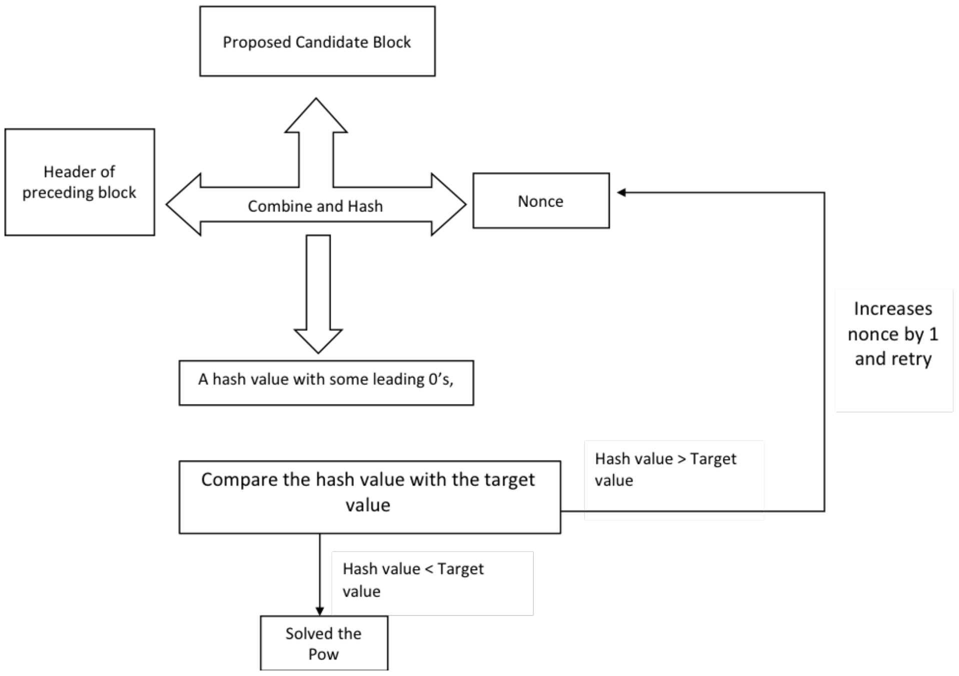

Proposed Candidate Block
Header of preceding block
Combine and Hash
Nonce
Increases nonce by 1 and retry
A hash value with some leading 0's,
Hash value > Target value
Compare the hash value with the target value
Hash value < Target value
Solved the Pow

FIGURE 3.7. The process of PoW algorithm.

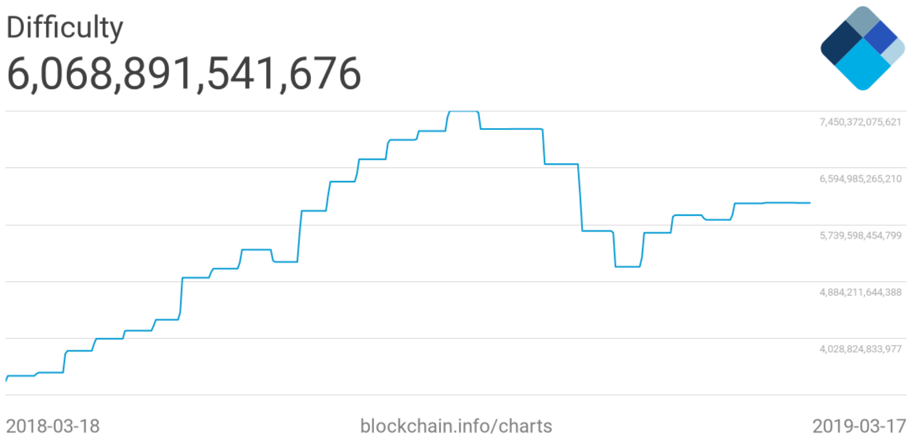


FIGURE 3.8. Changes in Bitcoin mining difficulty over a year.

3.2.3. Assembly of Blocks. To create a new block, mining nodes compete against each other by solving the PoW algorithm. Once a block is created, nodes will next broadcast this block across the network same as the transaction propagation. Nodes work collaboratively. They collect and verify the block, then attempt to link the validated block to the existing blockchain by finding its preceding block through the previous block hash. A transaction is confirmed when its associated block has connected to blockchain successfully. However, Bitcoin is a decentralized structure. Nodes might receive the blocks at a different time, and the ledger or the chain that they maintain might not always be consistent. In Bitcoin, nodes maintain three sets of chains [**3**].

1) The *main* chain. The blockchain with the most cumulative difficulty; normally this is also the chain with the greatest height.
2) The *secondary* chain. The branch of the main blockchain.
3) The *orphan* chain. It is the set of blocks that could not find its preceding block in the known chains.

Nodes always select the main chain which is the chain possesses the most cumulative difficulty to extend, and normally this chain contains the most of the blocks. However, sometimes the preceding blocks might not list in the main chain, under this circumstance, nodes will extend the secondary chain and then compare its cumulative difficulty to the main's. If the secondary chain possesses the greatest difficulty, then the secondary will be selected as the main chain. As long as the system follows this selecting mechanism, Bitcoin eventually reaches the consensus. For more details on block aggregation, see [**3**, **30**].

3.3. Blockchain Forks

Solving the PoW requires significant computational effort in a timely manner. Once a new block has been transmitted, the miners will start to mine the next new block. When there are multiple mining nodes to mine blocks for rewards, multiple blocks could be found at the same time, and this is called the *fork* event.

Blockchain forks occur due to the versions of chains inconsistency, it happens when there are multiple blocks competing to form the main chain [**3**]. It will be resolved automatically as nodes will converge on the main chain [**3**, **6**], this is also classified as *accidental forks.* The programming forks are classified as *Intentional forks* [**1**]. In this thesis, we only focus on accidental forks.

3.4. Models for blockchain forking

In order to understand how peers choose the chains in the fork events, we next use two novel models to simulate the progress, which are implemented in Matlab [**43**]. For simplicity, we assume the following parameters are given. First, the number of mining nodes during the mining process is fixed to a positive integer K. Second, the height of the blockchain is fixed to a positive integer N.

We named the first model as the FORK_1 model, since it captures the fork events without calculating the cumulative difficulty of the chains. We now describe the model.

1) At each level, every mining node is assigned a uniformly distributed random number from an interval $[0, 1]$. This represents the effort a mining node has contributed to mining.
2) Peers are choosing to connect the block with the highest effort.
3) The model outputs the corresponding array with respect to the miners' positions.

For example, suppose there are three mining nodes Miner_1, Miner_2 and Miner_3 working in the Bitcoin network. At each level, without loss of generality three blocks could be found simultaneously. By running FORK_1 with $K = 3$ and $N = 3$, we obtain the following table. Each entry represents how much computing power has been deployed by the miner. At level 1, Miner_3's block has the highest computing power, peers choose this block to extend. Similarly, peers will choose the blocks of Miner_1 and Miner_3 at level 2 and 3, respectively.

TABLE 3.1. An example of the FORK_1 model.

	Miner_1	Miner_2	Miner_3
N=1	0.9649	0.1576	0.9706
N=2	0.9572	0.4854	0.8003
N=3	0.1419	0.4218	0.9157

The model will then output the miners' position $[3, 1, 3]$. However, this model is impractical, since the target values and the cumulative difficulty still need to be considered.

The second model is called FORK$_2$, and it takes the cumulative difficulty into consideration. Let's assume there are multiple candidate blocks competing in the network, and nodes might select any of these blocks to extend. After the first round selection, bloackchain split into two chains. We label the main chain as of left, and the secondary as of right. The model will output a sequence of chain selections. We now describe the FORK$_2$ model.

1) Each chain is assigned a uniformly distributed random number from an interval $[0, 1]$ represents its difficulty at each level.
2) Calculate the respective cumulative difficulty.
3) At each level, a new block will be connected to the chain with the greatest cumulative difficulty.
4) The model output the corresponding sequence of chain selections.

Under the same assumptions, three miners work simultaneously. Instead of calculating miners' effort at each level, we measure the chains' cumulative difficulty. The results are shown below.

	$N = 1$	$N = 2$	$N = 3$
Main chain difficulty	0.7094	0.7547	0.1626
Secondary chain difficulty	0.6797	0.6551	0.3050
Main chain cumulative difficulty	0.7094	1.4641	1.6267
Secondary chain cumulative difficulty	0.6797	1.3348	1.6398

$$\text{Output : } \quad [\text{left}, \text{left}, \text{right}]$$

TABLE 3.2. Results of the model.

At $N = 1$, peers choose the main chain which is labeled as of left to extend, since it obtains the highest cumulative difficulty. At $N = 2$, the difficulty of the main chain is 0.7547, and the cumulative difficulty of the main chain is computed by adding level 2's difficulty to level 1's, which equals to 1.4641. The main chain still carries the most cumulative difficulty at $N = 2$, so this chain will be selected to extend. At $N = 3$, peers will converge on the secondary chain which is labeled as of right, since its cumulative difficulty exceeds the main chain. The model outputs the sequence of selections $[\text{left}, \text{left}, \text{right}]$. In reality, the cumulative difficulty can not be reprocessed, and the data will be much complicated than we consumed. Moreover, the number of miners is a random number at each level, the mining difficulty will be affected in order to maintain the ideal average mining time. For more background on the forks, the readers are directed to [**1**, **3**, **6**, **47**].

CHAPTER 4

Bitcoin Protocols

4.1. Introduction

As we described in the previous chapters, the decentralization of Bitcoin is the core design principle of the network [**3**]. The Bitcoin architecture is a *peer-to-peer* (P2P) structure. All the participants are called *peers*. Peers cooperate with each other to accomplish the processes of data storage, data transmission, data validation and data confirmation. Transactions are propagated by peers, and confirmed by recording in blockchain successfully through the mining process [**25**]. Moreover, all the confirmed transactions are also verifiable under P2P architecture.

The *Bitcoin network* is a network with nodes running the P2P protocol [**3**]. The key properties of the system are its scalability and dynamic nature [**36**]. To better interpret the P2P system, we need to understand the P2P network formation.

4.2. Bitcoin Network

We may consider the Bitcoin network as a graph $G = (V(G), E(G))$, where the set of nodes $V(G)$ represents a collection of nodes which are running the P2P protocol, and the set of edges $E(G)$ represents the connections between the nodes. For example, if there is a connection between two nodes, then they form an edge. One of such graphical representations is shown below. Figure 4.1 represents a toy network with only five participants, but the actual network will be much more complicated and dynamic in reality, as the connections may change over time.

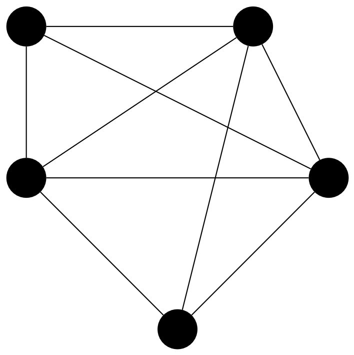


FIGURE 4.1. A Peer-to-Peer network.

When a new peer joins the network, a new node appears in the graph G, denoted by node A in Figure 4.2. In order to participate into the network, node A must establish connections with at least one existing node in the network, which means the corresponding nodes become adjacent. However, the network topology is not geographically defined; that is the choices of connections can vary [**3**, **56**]. Although existing nodes can be selected uniformly, there exist some long-running nodes called *seed nodes* [**3**]. Seed nodes provide a list of nodes of the network, and they can be utilized by discovering other nodes in the network quickly [**3**, **55**].

Seed nodes can be considered as a cluster in the network, but the connections from the seed nodes to others are sustained as inactive; that is, the connections from the seed nodes to others vanish in the network as the nodes will create their own potential contact lists. Once connections are established, node A will start to send a message which includes its IP address to its peers. The peers will then forward this message to their peers [**3**]. A distributed network is created and changed by peers, it is necessary for a participating node to establish various connections continuously.

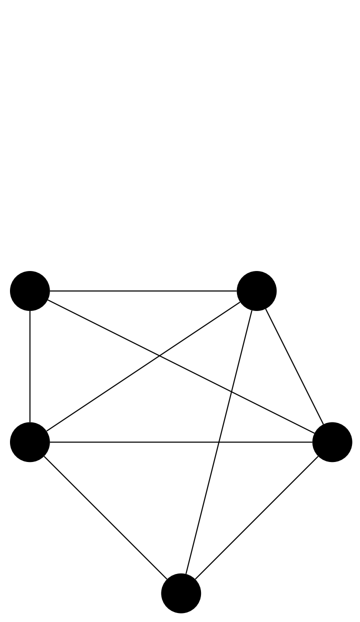


FIGURE 4.2. A graph G with new node A.

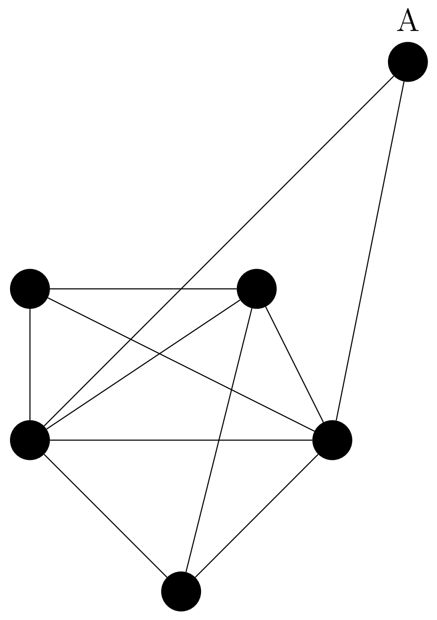


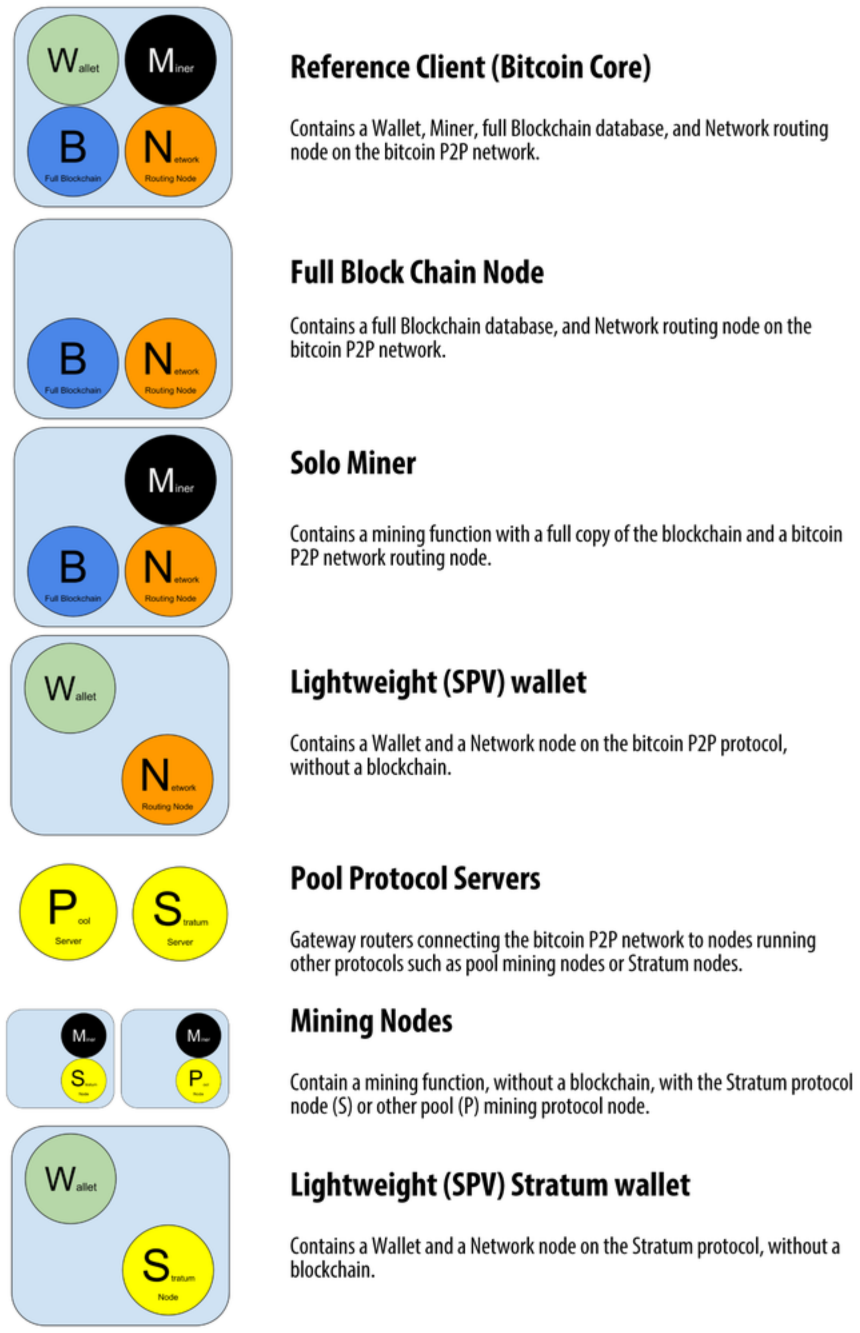
FIGURE 4.3. Node A established new connections.

Based on the formation of the P2P system, we observe that the Bitcoin network is a complex network. The Bitcoin network exhibits two major properties of complex networks: they are *large scale* and *dynamic* [**11**]. In [**17**,**49**], the Bitcoin users graph and the transactions graph were studied. These studies showed that both graphs also exhibit other complex network properties. For more details in the P2P network, readers are directed to [**3**, **5**, **25**, **36**, **55**, **56**].

P2P is the main protocol used in the Bitcoin network, but it is not the only protocol. There are additional protocols for different peers. The Bitcoin network with additional protocols refers to the *extended Bitcoin network*. As we described before, peers play similar roles over the distributed network. The distinction between them is their functionality. There are four main functionalities for a bitcoin node [**3**].

1) *Routing* service. Routing service is the required functionality for a node to participate in the P2P network. It enables peers to transmit and validate transactions.
2) The *blockchain database* service. A node with a completed database in the sense that they can autonomously verify any transactions without other resources needed.
3) *Mining* service. A node with mining service is responsible for creating new blocks through the mining process.
4) *Wallet* service. Nodes with wallet service are the nodes running on smartphones devices, or other devices with limited processing capabilities.

Moreover, there are other services over the network that indicate various protocols. A node with all four functionalities is called the *full node*. A full node is the most capable node in the network. It not only helps the network in the validation, but also in the mining processes. There are some nodes without routing service running other protocols in the extended bitcoin network, such as the *Stratum* protocol. These protocols are always used for mining purposes and for accessing into the Bitcoin network. The figures below are adapted from [**3**]. Figure 4.4 lists the types of nodes in the extended Bitcoin network, and Figure 4.5 is the geometrical representation of the extended Bitcoin network. For more details on the bitcoin protocol, see [**3**].



Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.

Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.

Solo Miner

Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.

Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.

Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.

Mining Nodes

Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.

Lightweight (SPV) Stratum wallet

Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.

FIGURE 4.4. A list of types of nodes in the extended Bitcoin network.

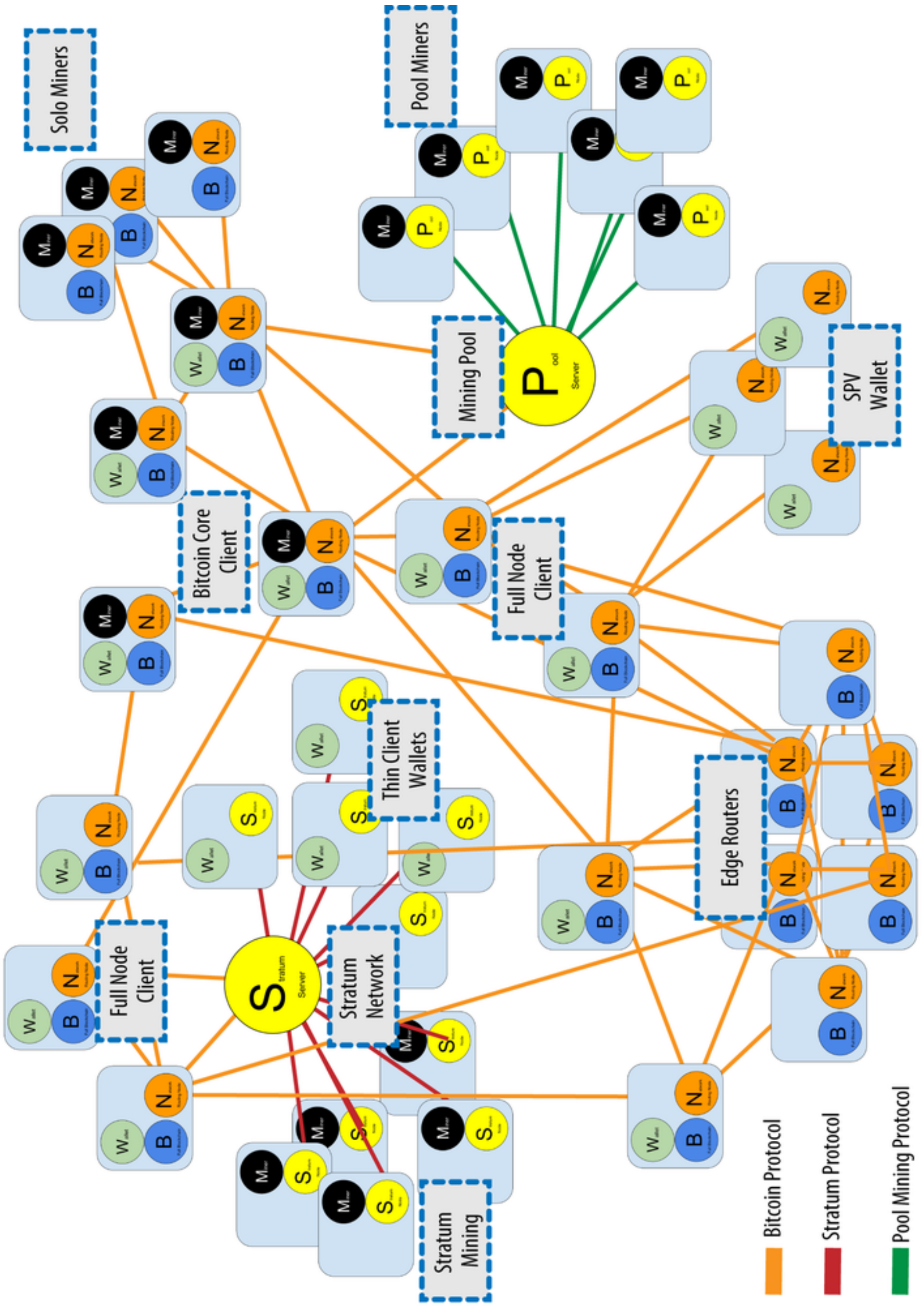


Figure 4.5. An extended Bitcoin network.

4.3. A model for Bitcoin protocols

The extended Bitcoin network consists of a collection of different types of nodes and the connections between these nodes over time. There is always a path from these additional protocols to the P2P main protocol. If we consider Figure 4.5, with the extended Bitcoin network as a graph $G = (V(G), E(G))$, then the nodes running with P2P protocol can be viewed as the subgraph of G. We present a new mathematical model to simulate these protocol graphs, and the model is implemented in Matlab [**43**].

We consider the following four parameters to simulate the network. First, the number of nodes in the network at the initial time $T = 0$ is a positive integer N. Second, the processing time is fixed to a positive integer t. Third, the number of types of nodes is a positive integer r with $r \geq 4$, since nodes differ from their services. The last parameter is a predetermined fixed probability p which is used as the criteria to determine whether to generate a new node or not. This model is named as the Random$_{protocol}$ model. We next describe the model.

1) At $T = 0$ there are N nodes, denoted by $V_1, V_2, \cdots, V_N$. These nodes form a path P_N, so there are $\mathrm{N} - 1$ edges. The model generates a clique K_m, where $m \in \{1, 2, 3, \cdots, r\}$ for each node V_i. The order of the clique K_m represents the type of that node.
2) Preform the following steps Nt times independently.

i) For each node V_i generate a probability p_{ij} at time $T = j$, where $j = 1, \cdots, t$.

ii) If p_{ij} is greater than p, then the model will generate a new node V_{ij}, and a clique K_m where $m \in \{1, 2, 3, \cdots, r\}$ for node V_{ij}. There is a new path from V_i to V_{ij}.

iii) If p_{ij} is less than p, then no new nodes or edges are added.

3) Finally, the model outputs a $(t + 1) \times N$ matrix. Each entry V_{ij} represents the connection changes of the node V_i at time t.

For example, consider the following choice of parameters: $n = 6, t = 3, r = 6$, and $p = 0.34632$ as inputs. A simulation gives the following matrix.

$$\begin{bmatrix} 2 & 6 & 6 & 3 & 3 & 3 \\ 0 & 1 & 2 & 5 & 1 & 6 \\ 2 & 2 & 0 & 5 & 1 & 6 \\ 5 & 0 & 3 & 0 & 5 & 6 \end{bmatrix}$$

A matrix representation of the Random$_{protocol}$ model.

We now interpret the resulting matrix. The column represents the connections of nodes in the extended Bitcoin network, and the row represents nodes at time $T = j$. The number of each entry describes the clique that the node has at time $T = j$. If we have zero in the entry, then there is no new node generation at that time. The matrix is visualized in Figure 4.6.

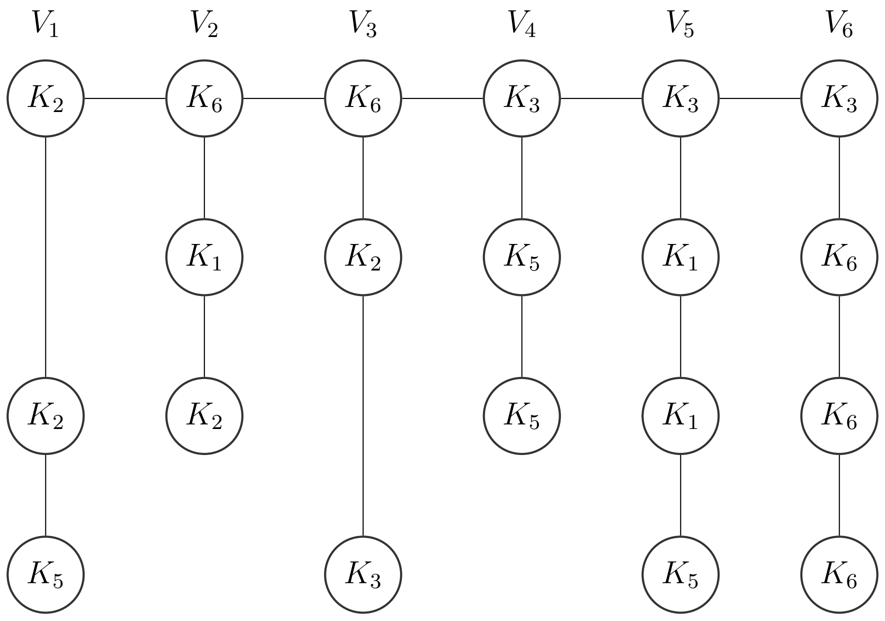


FIGURE 4.6. A visualization of the simulation.

From the description of the model, we note that as long as p_{ij} is greater than p, the model will generate a new node. The parameter p is the determinant of having a new node at time t. If we consider the order and the size of the resulted graph G as random variables, then we can calculate the corresponding expected values in general.

4.3.1. Expected order and size. Recall from Chapter 1, that the expected value of a random variable $\mathcal{X}$ is defined as

$$\mathbb{E}(\mathcal{X}) = \textstyle\sum_{x \in \Omega} x \ \mathbb{P}(x),$$

where $\mathbb{P}(x)$ is the probability of event x occurring, and Ω is the sample space of $\mathcal{X}$. If we consider each entry of the simulation matrix as a node, then there are $N(t+1)$ nodes, and there is a clique K_m inside each node.

To calculate the expected order of G, we need to calculate the expected order of the cliques.

We define random variables M_r to be the order of cliques, $\mathcal{X}$ to be the order of the graph G and $\mathcal{Y}$ to be the size of the graph G. Since the order of a clique K_m inside each node is randomly selected from $\{1, 2, 3, \ldots, r\}$, the expected order of cliques is given by

$$\begin{aligned}\mathbb{E}(M_r) &= \sum_{m=1}^{r} m\frac{1}{r}\\ &= \frac{1}{r}\sum_{m=1}^{r} m\\ &= \frac{1}{r}\frac{(1+r)\ r}{2}\\ &= \frac{1+r}{2}.\end{aligned}$$

The expected order of G is then given by:

$$\mathbb{E}(\mathcal{X}) = (t+1)Np\frac{1+r}{2}.$$

Similarly, the expected size of G is defined as follows.

$$\mathbb{E}(\mathcal{Y}) = \textstyle\sum_{Y\in\Omega'} y\ \mathbb{P}(y).$$

Due to the linearity of the expectation, the expected size of G equals to the sum of the expected size of cliques, the expected size of a path at the initial time and the expected value of edges from nodes to the original path. We define the random variables M_s, P_s and E_s to be the size of cliques, size of a path at the initial time, and the number of edges from nodes to the original path, respectively. We then have that the expected size of G is given by

$$\mathbb{E}(\mathcal{Y}) = \mathbb{E}(M_s) + \mathbb{E}(P_s) + \mathbb{E}(E_s).$$

These three expectations are calculated as follows.

$$\mathbb{E}(M_s) = \binom{\frac{1+r}{2}}{2},$$

$$\mathbb{E}(P_s) = N - 1,$$

$$\mathbb{E}(E_s) = pNt.$$

Therefore, the expected size equals to

$$\mathbb{E}(\mathcal{Y}) = \binom{\frac{1+r}{2}}{2} + N(pt+1) - 1.$$

In conclusion, we note some drawbacks of the model. First, note that the Random$_{protocol}$ model simulates the scenario that there is a path between nodes even though they are running different protocols or services. However, a path may be too simple for a structure; further, the network is changing over time. Note that it is difficult to capture all of the network dynamics with only a few parameters, as the protocol network contains several unpredictable factors such as network anonymity. In addition, the model uses a fixed random value p as the only criteria to determine the nodes changes over the network. In practice, we would need to fine tune this value to create accurate simulations. Finally, there is no real data on Bitcoin protocol graphs that we could find at the time of writing, making it challenging to test the accuracy of the model.

CHAPTER 5

Conclusion and Open Problems

5.1. Conclusion

In this thesis, we introduced and described the mechanics of Bitcoin and blockchain. We described the terminology of graph theory and probability theory in Chapter 1. In Chapter 2, we focused on the cryptographic foundations underpinning Bitcoin, and explained the mathematics behind it, such as hash and one-way functions. We also explained how elliptic curve multiplication is used in Bitcoin. In Chapter 3, we described how blockchain works when applied to Bitcoin, and presented two MATLAB models simulating the fork events during the mining process. We considered the probability as the accumulated difficulty, and examined the models with different sample data.

In Chapter 4, we considered the Bitcoin protocol graph. We also presented a MATLAB model to simulate how peers linked with each other on the protocol graphs. In addition, we calculated the expected values for the resulting graphs, and the results are stated below. The parameters N, t, r, p represent the number of nodes at initial time, the processing time, the number of types of nodes, and the random probability respectively.

i) The expected order of the resulting graph $\mathbb{E}(\mathcal{X})$ is given by

$$(t+1)Np\frac{1+r}{2}.$$

ii) The expected size of the resulting graph is then given by

$$\mathbb{E}(\mathcal{Y}) = \binom{\frac{1+r}{2}}{2} + N\ (pt+1) - 1.$$

5.2. Open Problems

The models we introduced served as tools to simulate events that emerge in the Bitcoin system. However, due to the lack of data and the complexity of the system, the models may be improved. In what follows, we state some open problems.

1. In Chapter 3, we simulated the fork events by considering the number of miners and their mining difficulty. As we stated when describing the drawbacks of models, the accumulated difficulty can not be reprocessed, and it is related to the ideal mining time. What is the outcome if we consider the relationship between the mining difficulties and ideal mining time in the model? Does there exist an appropriate model to approach the mining difficulty?
2. As we described in Chapter 4, the protocol graph is large-scale and has dynamic properties. Does it exhibit the other properties of complex networks, such as the small world property and power law degree distributions? What variables do we need to employ in

the model to better approximate reality? Moreover, the expected size of the cliques is calculated by $\binom{\frac{1+r}{2}}{2}$; how would the result change if $\frac{1+r}{2}$ is not an integer value?

3. The blockchain technology provides the public storage of data, meaning that every user in the network can access the information that is stored in the database. In this thesis, simulated these systems using stochastic models. If actual data was used, then we could fine tune the model, and we plan on doing this in future work.
4. As we stated in Chapter 4, each node has its own potential connection list. By considering the connection between each other as an edge, and each peer as a node, we can obtain a graph with millions of nodes and edges. Can we use the *Iterated Local Transitivity* (ILT) model [**10**] to simulate the system?

Appendix

Elliptic Curve Multiplication Over Finite Field

We present the MATLAB code discussed in Chapter 2.

```
% An elliptic curve equation y^2=x^3- x+10 (mod p)
% over finite field with order p
p=23;            %Prime number
x=[];
y=[];
q=[0:p-1];
for i=1:p
b=mod(q(i)^2,p);
a=mod(q(i)^3-q(i)+10,p);
y=[y,b];
x=[x,a];
end

% Define the resulted set as S, it is an empty set
    at the beginning.
```

```
n=[];

%check if x^3-x+10 is square modulo of p

for i=1:numel(x)
      r=find(x(i)==y); % Use find() function to find
         the indices
      for j=1:numel(r)
      n=[n;q(i),q(r(j))];% Produce a set of points
      end
end

% Isolate the coordinates

x1=n(:,1);
y1=n(:,2);
plot(x1,y1,'bo')
set(gca,'XTICK',0:p);
set(gca,'YTICK',0:p);
grid on;
grid minor;
```

Models for Blockchain Forking

We now present the code discussed in Chapter 3.

Listing 5.1. $Fork_1$ Model.

```

% N represents the height(time) of the block, it is
    fixed and positive integer
% K represents the number of miners during the
   mining process, it's fixed and positive interger

function Fork_1(K,N)
% x is the set of chosen miner
% y is the set of possitions
K=3
N=3
% Use for loop to construct the matrix, and produce
    x and y
for i=1:N
    for j=1:K
```

```
        m(i,j)=rand; % A single uniformly distributed
            random number in the interval (0,1).
        end
        [x(i),y(i)]=max(m(i,:));
end
m
Y=y % The array of miniers' positon
```

Listing 5.2. *Fork*$_2$ Model.

```

function FORK_2(K,N)
K=3;
N=3;
 diff_left=rand(1,N)
 diff_right=rand(1,N)

% using for loop to obtain the cumulative difficulty
      for the left and right chain
 for i=1:N
      left(i)=sum(diff_left(1:i))
```

```
    right(i)=sum(diff_right(1:i))
    if left(i)==right(i)
        diff_left(i)=rand;
        diff_right(i)=rand;
        left(i)=sum(diff_left(1:i));
        right(i)=sum(diff_right(1:i));
    end
end

% using for loop to produce the set of chosen miners
    and their corresponding position
for i=1:N
    for j=1:K
   m(i,j)=rand;
    end
    [x(i),y(i)]=max(m(i,:));
end


seq=strsplit(num2str(y));
for i=1:N
```

```
        if left(i)>right(i)
            seq(i)={'left'};
        else
            seq(i)={'right'};
        end
end
seq     % The sequence of chains
left    % the left(main) cumulative difficulty chain
right   % the right(secondary) cumulative difficulty
    chain
y       % The arrary of miners' position
```

Model for Bitcoin Protocols

Listing 5.3. Random$_{protocol}$ Model.

```

function Random_protocol(N,t,r)
N=6
t=3
r=6
p=rand % We used p= 0.

```

```
m=ones(t+1,N);
 for i=1:N
  m(1,i)=randsample(r,1);
 end
 for i=2:t+1
        for j=1:N
                q=rand;
                if (q < p)
                        m(i,j)=0;
                else
                        m(i,j)=randsample(r,1);
                end
        end
 end
m % Produce the result.
```

Bibliography

[1] N. Acheson, Hard Fork vs Soft Fork, Retrieved March 18, 2019, from `https://bit.ly/2UhlMQz`.

[2] V.S. Adamchik, Binary Tree, Retrieved March 16, 2019, from `https://www.cs.cmu.edu/~adamchik/15-121/lectures/Trees/trees.html`.

[3] A.M. Antonopoulos, *Mastering bitcoin: unlocking digital cryptocurrencies*, O'Reilly Media, Inc., 2014.

[4] A. Back, Hashcash-a denial of service counter-measure, Retrieved March 16, 2019, from `http://www.hashcash.org/papers/hashcash.pdf`.

[5] H. Balakrishnan, M.F. Kaashoek, D. Karger, R. Morris, I. Stoica, Looking up data in P2P systems, *Communications of the ACM* **46(2)** (2003) 43-48.

[6] A. Baliga, *Understanding blockchain consensus models*, Persistent, 2017.

[7] Bitcoin Crypto-Economics Index by CoinDesk, Retrieved February 05, 2019, from `https://www.coindesk.com/price/bitcoin`.

[8] Bitcoin Difficulty, Retrieved March 17, 2019, from `https://bit.ly/2OCKE3y`.

[9] A. Bonato, *A Course on the Web Graph*, American Mathematical Society Graduate Studies Series in Mathematics, 2008.

[10] A. Bonato, N. Hadi, P. Horn, P. Prałat, C. Wang, Models of online social networks, *Internet Mathematics* **6** (2009) 285-313.

[11] A. Bonato, A. Tian, Complex networks and social networks, *Social Networks*, Springer 2011 pp. 280-291.

[12] G. Brassard, *Modern cryptology: A tutorial, Volume **325** of LNCS*, Springer, 1988.

[13] A. Canteaut, F.L.D.V.G. Norton, *Modern cryptology*, `https://bit.ly/2T2QNeC`.

[14] Connecting the world to Crypto, Retrieved March 18, 2019, from `https://www.blockchain.com/`.

[15] A. Corbellin, Elliptic Curve Cryptography: A gentle introduction, Retrieved February 16, 2019, from `https://bit.ly/2EF7sMz`.

[16] M. Crosby, P. Pattanayak, S. Verma, V. Kalyanaraman, Blockchain technology: beyond bitcoin, *Applied Innovation* **2(6-10)** (2016) 71.

[17] I. Csabai, D. Kondor, M. Pósfai, G. Vattay, Do the rich get richer? An empirical analysis of the Bitcoin transaction network, *PLOS ONE* **9(2)** (2014) e86197.

[18] B. Curran, What is a Merkle Tree? Beginner's Guide to this Blockchain Component, Retrieved March 16, 2019, from `https://blockonomi.com/merkle-tree/`.

[19] L. Coleman, Big Four Giant PwC Announces Blockchain Auditing Service, Retrieved February 6, 2019, from `https://bit.ly/2ppJIDp`.

[20] S. Das, 'Big Four' Giant Deloitte Completes Successful Blockchain Audit, Retrieved February 1, 2019, form `https://bit.ly/2USbTJj`.

[21] S. David, G. Geoffrey, *Probability and Random Processes*, Oxford University Press, 2001.

[22] Desmos graph, Retrieved February 16, 2019, from `https://www.desmos.com/calculator`.

[23] Difficulty, Retrieved March 25,2019, from `https://en.bitcoin.it/wiki/Difficulty`.

[24] W. Diffie, M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory* **22** (1976) 644–665.

[25] J.A.D. Donet, J. Herrera-Joancomartí, C. Pérez-Sola, The bitcoin P2P network, In: *International Conference on Financial Cryptography and Data Security*, Springer 2014 pp. 87-102.

[26] D.S. Dummit, R.M. Foote. *Abstract algebra Volume 3*. John Wiley & Sons, 2004.

[27] C. Dwork, M. Naor, Pricing via Processing or Combatting Junk Mail, In: *Annual International Cryptology Conference*, Springer 1992 pp. 139-147.

[28] Elliptic Curve Points, Retrieved February 16, 2019, from `https://www.desmos.com/calculator/ialhd71we3`.

[29] Facebook users worldwide 2018, Retrieved February 1, 2019, from `https://bit.ly/2daz7Yr`.

[30] E. Feig, A Framework for Blockchain-Based Applications, *ArXiv preprint* (2018) arXiv:1803.00892.

[31] J. Frankenfield, *Block Header (Cryptocurrency)*. Retrieved March 16, 2019, from `https://bit.ly/2Wc9TMm`.

[32] S.W. Gebregiyorgis, *Algorithms for the elliptic curve discrete logarithm and the approximate common divisor problem*, PhD Thesis, University of Auckland, 2016.

[33] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge University Press, 2009.

[34] M. Gray, Ethereum Blockchain as a service now on Azure. Retrieved February 6, 2019, from `https://bit.ly/2TB5YYK`.

[35] D. Hankerson, A.J. Menezes, S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[36] M. Jelasity, A. Montresor, PeerSim: A scalable P2P simulator, In: *2009 IEEE Ninth International Conference on Peer-to-Peer Computing* (2009) 99-100.

[37] E. Kozliner, *Merkle Tree Introduction*, Retrieved March 16, 2019, from `https://bit.ly/2Y7qYsN`.

[38] R. Lidl, H. Niederreiter, *Finite Fields, Second Edition*, Cambridge University Press, 1997.

[39] Linux Foundation Unites Leaders to Advance Blockchain Technology, Retrieved February 10, 2019 from `https://ibm.co/2O2CU7W`

[40] D. Liu, L.J. Camp, Proof of Work can Work, In: *5th Annual Workshop on the Economics of Information Security*, Cambridge 2006.

[41] J.C. Lubbe, J.C. Van Der Lubbe, *Basic Methods of Cryptography*, Cambridge University Press, 1998.

[42] Mario, Bitcoin history, Retrieved February 1, 2019, from `https://bit.ly/2UNpG3J`

[43] MATLAB and Statistics Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States.

[44] R. Merkle, *Secrecy, authentication and public key systems/ A certified digital signature*, Ph.D. dissertation, Stanford University, 1979.

[45] Merkle Tree, Retrieved March 15, 2019, from `https://docs.neo.org/developerguide/en/articles/cryptography/merkle_tree.html`.

[46] T. Murphy, *Course 373-Finite Fields*, University of Dublin, Trinity College School of Mathematics, `http://www.maths.tcd.ie/pub/Maths/Courseware/FiniteFields/FiniteFields.pdf`, 2006

[47] N. Satoshi, Bitcoin: A peer-to-peer electronic cash system. Retrieved from `http://www.bitcoin.org/bitcoin.pdf`.

[48] Protocol rules, Retrieved February 16, 2019, from `https://en.bitcoin.it/wiki/Protocol_rules`.

[49] L. Ricci, D.D.F. Maesa, A. Marino, Data-driven analysis of Bitcoin properties: exploiting the users graph, *International Journal of Data Science and Analytics* **6(1)** (2018) 63-80.

[50] A. Sawlikar, Point Multiplication Methods for Elliptic curve Cryptography, *International Journal of Engineering and Innovative Technology (IJEIT)* **1.1** (2012) 1-4.

[51] Script, Retrieved February 18, 2019, from `https://en.bitcoin.it/wiki/Script`.

[52] Secp256k1, Retrieved February 16, 2019, from `https://en.bitcoin.it/wiki/Secp256k1`.

[53] J.H. Silverman, *An introduction to the theory of elliptic curves*, Brown University and NTRU Cryptosystems, Inc., `https://www.math.brown.edu/~jhs/Presentations/WyomingEllipticCurve.pdf`, 2006

[54] S.H. Strogatz, D.J. Watts, Collective dynamics of 'small-world' networks, *Nature* **393** (1998) 440-442.

[55] Tendermint P2P Layer, Retrieved April 12, 2019 from `https://kb.certus.one/peers.html`.

[56] S. Tochner, A. Zohar, How to pick your friends-a game theoretic approach to p2p overlay construction, *ArXiv preprint* (2018) arXiv:1810.05447.

[57] D.B. West, *Introduction to Graph Theory, 2nd edition*, Prentice Hall, 2001.