Modern Cryptography Techniques and Cryptocurrencies

BY

Brian MacCarvill

2022/2023 Project report submitted in partial fulfilment of the examination requirements leading to the award of
BSc in Mathematical Sciences
Technological University Dublin

Supervised by

Paul Molloy

Contents

1 Introduction

The aim of this project is to outline and examine a brief history on cryptography, modern cryptography techniques and the cryptography behind cryptocurrencies all from a mathematical perspective.

There are a myriad of different reasons that someone would want to keep a message or a piece of sensitive information secret and encrypting this information is sometimes the best way to do it. There are also a myriad of reasons that a third party would want to decrypt the contents of a sensitive file or the real meaning behind a message. This on many occasions has led to an arms race of sorts, in which one party tries to find more elaborate and efficient ways to encrypt something while another party tries to decrypt ciphertext in more sophisticated and more efficient ways.

Cryptography and the continuous improvement of encryption methods versus the continuous improvements in decryption methods is playing a very important role in the modern world.

Internet security, people's email, social media, and phone passwords are hidden behind a wall of encryption and people are dedicating more electricity than Argentina to mining cryptocurrencies (Criddle, 2021). Cryptography is playing an ever-increasing role in our society.

The aim of this project is to

- Outline a brief description of historical ciphers and the evolution of cryptography through history.
- Give a detailed explanation of modern cryptographic techniques particularly RSA and Elliptic Curves.
- Present a detailed explanation of the encryption methods behind cryptocurrencies.

2 Historic Cryptographic Techniques

The earliest evidence of cryptography we have was in ancient Egypt in 1900bc where scribes used hieroglyphs in a non-standard fashion (Damico, 2009). Since then (and likely earlier) cryptography has been used in different ways for many reason. Caesar Augustus used shift ciphers to cummunicate with his generals. Mary Queen of Scots used shift and substitution ciphers while held in captivity.

2.1 Frequency of Letters

Letters in every language are not used equally. In every language some letters are far more common than others and some sequences of letters are far more common than others. It is helpful to know this when decrypting ciphertext. If patterns in the frequency of letters in a ciphertext can be seen this can give crucial insight to the cipher, and you can match it to the word frequency of your chosen language. In English the frequency of letters is

letter	frequency	letter	frequency
a	0.08167	n	0.06749
b	0.01492	o	0.07507
c	0.02782	p	0.01929
d	0.04253	q	0.00095
e	0.12702	r	0.05987
f	0.02228	s	0.06327
g	0.02015	t	0.09056
h	0.06094	u	0.02758
i	0.06966	v	0.00978
j	0.00153	w	0.02360
k	0.00772	x	0.00150
l	0.04025	y	0.01974
m	0.02406	z	0.00074

The most common combinations of 2 letters or bigrams are

he, in, en, nt, re, er, an, ti, es, on, at, se, nd, or, ar, al, te, co, de, to, ra, et, ed, it, sa, em, ro.

The most common combinations of 3 words or trigrams are

the and, tha, ent, ing, ion, tio, for, nde, has, nce, edt, tis, oft, sth, men.

2.2 Caeser Cipher/ Shift Cipher

Encryption in a shift cipher is performed by changing every letter in a message with the letter a certain distance away from the original letter. For example, to encrypt the word "brian" using the key 10, the original letters would turn into "laskx" While Caesar's cipher was the first recorded use of this scheme, other substitution ciphers are known to have been used earlier.

2.3 Substitution Cipher

Substitution cipher is created when the alphabet is scrambled and then every letter in the plaintext is replaced with the corresponding letter in the scrambled alphabet. This cipher

can't be broken by brute force because the number of ways and alphabet can be scrambled is 26!

Original alphabet abcdefghijklmnopqrstuvwxyz

Scrambled alphabet uarxzedfkwnbpmhvqscoitygjl

This is where word frequency is useful. For a simple plaintext like "brian" under this substitution cipher the plain text turns into the cipher text "askum". For larger texts the frequency of letters enables people to replace the substituted letters with the original letter.

A substitution cipher is almost impossible to decrypt given a small ciphertext using brute force because of the large number of potential keys. However substitution ciphers are easily broken if you look at word frequencies, if given a long enough text the frequencies of different letters become apparent. For example, using a long enough ciphertext and the substitution alphabet given above the letter c which corresponds to the letter t would appear roughly 9.056% of the time.

A substitution cipher can also be performed by using more than one scrambled alphabet.

Eg

Normal abcdefghijklmnopqrstuvwxyz

Scramble one neimkqhaorpljgyftvucbxwzds

Scramble two qmjvzapdowcnutiyskrehgfblx

Encrypt every odd number with first scrambled alphabet and every even number with second scrambled alphabet eg. "hello" becomes "azlny. With 2 scrambled alphabets the number of different keys is

$$26!^2$$

With n scrabbled alphabets the number of keys is

$$26!^n$$

2.4 Rotor encryption

With the advent of better technology in the 20th century, people saw that it could be used to make more complex encryption easier and quicker. The first ever mechanical encryption (in 1920s) took a substitution cypher and rotated the scrambled alphabet each letter.

One of the bases for new forms of encryption was the rotor machine. This machine was used in the German's "enigma machine" and the Japanese "purple" machine during WW2. The rotor machine is made of one or more shifting rotors. When a letter is typed/inputted the rotor machine outputs a different letter based on a scrambled alphabet.

Example, to code the word 'dog' we use this substation for the letter "d"

Normal abcdefghijklmnopqrstuvwxyz

Scramble neimkqhaorpljgyftvucbxwzds

Then every letter is shifted in the scrambled alphabet and then the second letter "o" is encrypted with the new alphabet.

Normal abcdefghijklmnopqrstuvwxyz

Scramble eimkqhaorpljgyftvucbxwzdsn

And do the same with the third letter "g"

Normal abcdefghijklmnopqrstuvwxyz

Scramble imkqhaorpljgyftvucbxwzdsne

So "dog" would become "mfo".

For rotor machines with more then one rotor, after the first letter is typed only the first rotor rotates. After the first rotor has completed a full rotation does the second rotor start to rotate and the first rotor stops rotating. The nth rotor starts to rotate only after the (n-1)th rotor has completed a full rotation.

Since the same letter written in difference places will most likely produce two different outputs, this makes looking at letter frequencies, bigrams, or trigrams ineffective in trying to decrypt the message. A rotor machine is also practically impossible to break by brute force because the number of possible keys for just one rotor is $26 * 26!$.

2.5 The Enigma Machine

One of the most famous mechanical encryption devices was the enigma machine build by the Germans during WW2. The enigma machine looks like a blocky typewriter. It is comprised of

- Two alphabets, one to write your plain text and one just with lights under each letter which lights up to signify the encrypted letter
- One plug board which simply swaps letters eg if $A \to B$ then $B \to A$
- 3 routers that are made up of scrambled up alphabets that all lead into each other. Later on there were more rotors used.
- A reflector that swaps letter for a second time.

The plug board is where letters are swapped, for example if you pressed A on the keyboard the plug board could swap it to B and if you pressed B it could swap it to A. In the real enigma machine, some letters are swapped, and some letters are not swapped depending on the plug-board set up.

A plug board takes an input and swaps it with a different letter, for example if A is the input and B is the output on the plugboard, then if B is the input A would be the output.

After the letter has been swapped in the plugboard, the new letter is put into the first rotor and a new letter is outputted. The letter outputted by the first rotor becomes the input for the second rotor, where another letter is outputted. Finally, the same process is repeated in the 3rd rotor, where the output of the second rotor is the input to the 3rd. But where does this output go?

The output of the final rotor then reaches the reflector, which like the plugboard just swaps letters, so for example if the input was T, then output might be J and if the input was J the output would be T. Like in the plug board some numbers might reflect to themselves in the reflector, eg $F \to F$.

Now go in reverse to get the encrypted letter.

The output of the reflector turns into the input for the third rotor and the process repeats

itself in reverse, where the output from the third rotor turns into the input for the second and the output from the second rotor turns into the input for the first and finally all the way back through the plugboard where the output lights up the encrypted letter on the machine.

What makes the enigma machine special is that after every letter the first rotor rotates by one letter, so pressing the same letter 2 times in a row does not produce the same output. After 26 letters the first rotor locks in place and the second rotor starts to rotate.

During WW2, every month German Enigma machine operators were issued with a key sheet printed with the daily settings for their network. The settings would indicate which 3 of the 5 rotors to use, the order of the rotors, the initial settings of each rotor wheels and which letters to connect by plugging cables on the plug board.

When an encrypted message was received, an enigma machine set up in the same way as the machine used to do the encryption was used. The operator merely typed in the encrypted message and the original message was outputted.

Figure 1: Enigma machine

3 Modern Cryptography Basis

Computers serve a unique problem when it comes to cryptography. While programming a computer to hold a long conversation is very difficult, setting them up to do tasks like solving an enigma machine or finding the frequency of different letters is easy.

To explain modern Cryptography, one must first learn some basic mathematical concepts, namely

- The Euclidean algorithm
- The extended Euclidean algorithm
- Chinese Remainder theorem
- Fermat's little theorem
- Certain properties of prime numbers.

3.1 Euclidean Algorithm

This is a method for finding the Greatest Common Denominator between two integers. The GCD of 2 integers a and b is written as GCD(a,b)

Method: to find the GCD(a,b)

1. If

$$b > a$$

then swap a and b

2. Divide a by b and round downwards to find C, where C is the greatest integer in which

$$a \geq bC$$

3. if $r = 0$ then b is the GCD of a and b

How does this work?

If it is assumed that

$$GCD(a, b) = d$$

r is found such that

$$a = bC + r$$

this can be rewritten as

$$r = a - bC$$

If $r \neq 0$ then it is known that $d|r$ (r is divisible by d) this comes from the property. If $x|y$ and $x|z$ then $x|(ty + hz)$ For any integers t, h.

From this it is known

$$d|a$$

and

$$d|b$$

so

$$d|(a - bC)$$

or

$$d|r$$

now continue to remove a and replace it with b and replace b with r so the GCD looks like GCD(b,r) continue from step 2 until one of the numbers in the GCD divides into the other without a remainder. Repeat this process with GCD(b,r) The system of equation would look like

$$a = b * c_1 + r_1$$

$$b = r_1 * c_2 + r_2$$

$$r1 = r_2 * c_3 + r_3$$

$$r_{n-1} = r_n * c_n + 1 + 0$$

$$gcd(a, b) = r_n$$

If the highest common divisor between two integers is 1 then the pair is said to be relatively prime Example For the GCD(546,99)

$$546 = 99 * 5 + 51$$

$$99 = 51 * 1 + 48$$

$$51 = 48 * 1 + 3$$

$$48 = 3 * 16 + 0$$

The $GCD(546, 99) = 3$

3.2 Extended Euclidean Algorithm

The Extended Euclidean Algorithm (EEA) is an algorithm to compute integers x and y such that

$$ax + by = GCD(a, b)$$

given integers a and b. the existence of x and y is given by Bézout's lemma (which is outside the scope of our project). The EEA works by reversing the steps in the Euclidean algorithm, which finds the GCD of a and b, GCD(a,b). For the example To find x and y such that

$$546x + 99y = gcd(546, 99)$$

Start with the line

$$51 = 48 * 1 + 3$$

and rearrange it so

$$3 = 51–(48 * 1)$$

eq 1 Then take the line about that $99 = 51 * 1 + 48$ and rearrange it so

$$48 = 99–(51 * 1)$$

Now replace 48 in eq 1 with this equation in formula

$$3 = 51–((99–(51 * 1)) * 1)$$

Re-arrange grouping all the 51s together so

$$3 = 51 * 2–99$$

Repeat this process so

$$546 = 99 * 5 + 51$$

becomes

$$51 = 546–99 * 5$$

And

$$3 = (546–99 * 5) * 2–99$$

$$3 = 546 * 2–11 * 99$$

So $x = 2\ and\ y = -11$

3.3 The Chinese Remainder Theorem: Two Equations

Chinese Remainder Theorem (CRT) is one of the most useful results from number theory (Stalling, 2021, p. 278). CRT for two equations

$$X \equiv a_1\ mod\ m_1\ and\ X \equiv a_2\ mod\ m_2$$

where $GCD(m_1, m_2)\ = 1$

Is used to find X mod (M*N) that satisfies both equations given above. First compute

$$T \equiv M^{-1} mod\ m_2$$

Which is possible since it has been assumed that $gcd(N, M) = 1$

$$u \equiv (a_2 - a_1) * T\ mod\ m_2$$

The solution modulo $M * N$ is then given by

$$X = a_1 + u * m_1$$

To see this always works verify that

$$u \equiv (a_2 - a_1) * T \; mod \; m_2$$

$$X \; modM \equiv a_1 + u * Mmod \; m_1 \equiv a_1$$

$$Xmod \; m_2 \equiv a_1 + u * Mmod \; m_2$$

$$\equiv a_1 + (a_2–a_1) * T * Mmod \; m_2$$

$$\equiv a_1 + (a_2–a_1) * M^{-1} * Mmod \; m_2$$

$$\equiv a_1 \equiv (a_2–a_1) \; mod \; m_2$$

$$\equiv a_2$$

Example

$$X \equiv 6 \; mod \; 11 \; and \; X \equiv 7 \; mod \; 17$$

Write X such that

$$X = 7 + 17u$$

Where u is a natural number

$$7 + 17u \equiv 6 \; mod \; 11$$

$$6u \equiv 6–7 \; mod \; 11 \equiv 10 \; mod \; 11$$

$$6 - 1 \; mod \; 11 \equiv 2 \; mod \; 11$$

$$U = 2 * 10 \equiv 9 \; mod \; 11$$

$$\ldots$$

$$U \equiv 9$$

$$x = 17 * 9 + 7 = 160$$

3.4 The Chinese Remainder Theorem: The General Case:

Given a system of equations

$$X \equiv a_1 \; mod \; m_1$$

$$X \equiv a_2 \; mod \; m_2$$

$$\dots$$

$$X \equiv a_r \; mod \; m_r$$

Where for $\forall \; i, j$ the $GCD(m_i, m_j) = 1$. If we want to find X modulo M, where

$$M = \prod_{i=1}^{r} m_i$$

that satisfies the system of equations.

$$X \equiv a_i \; mod \; m_i$$

for $\forall i$ The CRT states that

$$X \; mod \; M \equiv \sum_{i=1}^{r} c_i * a_i$$

where

$$c_i = (M/m_i) * ((M/m_i)^{-1} \; mod \; m_i)$$

Example

Given the system of equations

$$X = 4 \; mod \; 9$$

$$X = 8 \; mod \; 11$$

$$X = 9 \; mod \; 14$$

Compute

$$M = 9 * 11 * 14 = 1386$$

$$M_1 = 1386/9 = 154, M_2 = 1386/11 = 126, M_3 = 1386/14 = 99$$

$$M_1^{-1} = 154^{-1} mod \; 9 = 1^{-1} mod \; 9 = 1$$

$$M_2^{-1} = 126^{-1} mod \; 11 = 5^{-1} mod \; 11 = 9$$

$$M_3^{-1} = 99^{-1} mod \; 14 = 1^{-1} mod \; 14 = 1$$

Then the solution is given by

$$X = \sum_{1}^{3} a_i * M_i * (M_i^{-1}) mod\ M$$

$$= 4 * 154 * 1 + 8 * 126 * 9 + 9 * 99 * 1\ mod\ 1386$$

$$= 877\ mod\ 1386$$

CRT is used to write large numbers in terms of tuples of smaller numbers in modulo (Stalling, 2021, p. 280). CRT is also used to decrypt RSA encryption with small public exponents (Smart, 2016, p. 304).

3.5 Euler Phi-Function

The Euler phi-function, denoted as $\theta(n)$, is the number of natural numbers less than n relatively prime to n. We can note that for any prime number p

$$\theta(p) = p - 1$$

This is because because a prime number is not divisible by any less number than it. We can also note that for any prime to the power of a constant.

$$\theta(p^c) = p^c - p^{c-1}$$

To see this, note that, the natural numbers less than p^c in which the $gcd(p^c, x) \neq 1$ are only the numbers that are divisible by p. There are p^{c-1} numbers less than p^c that are divisible by p. An important fact about Euler phi-function for RSA is

$$\theta(pq) = \theta(p) * \theta(q)$$

To see this, list the number of integers less than pq is the set $[1, 2 \ldots, pq - 1]$. Any integer that divides pq must be divisible by p or q, so the integers that are not relatively prime to pq are the set $[\ p,\ 2p,\ \ldots,\ (q–1)p\ ]$ and the set $[\ q,\ 2q\ \ldots\ (p–1)q\ ]$. Each set has q-1 and p-1 number of element. So

$$\theta(pq) = (pq–1) – ((q - 1) + (p–1))$$

$$= pq - (p + q) + 1$$

$$= (p–1) * (q–1)$$

$$= \theta(p) * \theta(q)$$

3.6 Fermat's Little Theorem

Fermat's little theorem states

$$a^{p-1} \equiv 1 \, mod \, p$$

for a prime number p and a integer a that is relatively prime to p.

3.7 Prime Numbers

For modern ciphers, large prime numbers are often needed. There is no simple and efficient way of finding large prime numbers (Stalling, 2021, p. 275). It has also been proven that there is an infinite number of prime numbers. A large basis of modern cryptography relies on the fact that if you multiply 2 large prime numbers together, separating the product into its factors is difficult.

If large prime numbers are multiplied together, it is difficult to find the prime factors of the resulting number. The most efficient way to find these factors is to divide the number by every prime number smaller than the square root of it. For example for the number 1651933 (1471 × 1123) would start by checking 1651933 *mod* 2, then 1651933 *mod* 3 ... 1651933 *mod* 1123. It is possible to skip the first few prime numbers if you assume the large number isn't the product of a prime integer times a small number like 2, 3 or 5 but after that there is no better way to find the prime factors. For computers the order of the operations gets larger exponentially with a linear increase in the prime factors.

For the product of the 1000th (7919) prime number and the 999th (7907) prime number, the product (62615533) would need to be divided by 998 different prime numbers to enable you to find the prime factors.

For example, the product of the 50th (229) and 51st (233) prime number is 53357, and the

product of the 100st (541) and 101th (547) prime number is 295927.

295927 is roughly 5.5 times bigger then 53357. It would take 5.5 times more operations to find if a prime number is a factor of 295927 than it would to find if a prime number is a factor of 53357. It is also necessary to divide 295927 by twice as many numbers.

3.8 Public and Private key encryption/ Asymmetric encryption

Single key encryption is the only one written about so far in this thesis and until the 1970s the only one in use. Asymmetric encryption is when the key to encrypt a message and decrypt a message are different. Private key encryption relies on the fact that brute force attacks on the public key are impractical.

4 RSA encryption

RSA is one of the most popular cryptosystems in use today (Vaudenay, 2005, p. 184) . To create the public and private keys for an RSA encryption you must start by picking 2 very large random prime numbers p, q and a number e that satisfies $GCD(e, (p-1)(q-1)) = 1$. $N = p * q$. Your public key encryption is (N,e).

To compute the private key, an integer d must be found such that

$$e * d = 1 mod((p-1)(q-1))$$

d is computed by using the Extended Euclidean algorithm. The private key is (d,p,q). RSA relies on the fact that N is a large enough number that separating it into its prime factors is impractical.

If someone wants to encrypt a message to send represented by the letter m. m must be less than N. Then by only using a public key they can encrypt the letter m as follows

$$m^e mod N = c$$

where c is the encrypted message. To decrypted we use the private key.

$$c^d mod N = m$$

From formats little theorem

$$a^{p-1} = 1 mod p$$

And

$$a^{q-1} = 1 mod q$$

So

$$a^{(p-1)(q-1)} = 1 mod N$$

we write

$$c^d = (m^e)^d$$

$$= m^{ed}$$

Since $e * d = 1 \ mod \ ((p-1)(q-1))$ as said up above, we can say $e * d = 1 + s(p-1)(q-1)$ where s is an integer. So

$$= m^{1+s(p-1)(q-1)}$$

$$= m \times m^{s(p-1)(q-1)}$$

Find

$$c^d mod N$$

$$m \times m^{s(p-1)(q-1)} \ mod \ N$$

$$= m \times 1 mod \ N$$

Since m is less than N we can remove then modulo.

$$c^d mod \ N = m$$

RSA encryption relies on the difficulty of finding d given only the public key assuming a sufficiently large public and private key(Smart, 2016, p. 294). The European Agency for Network and Security Information recommend a key length (the value of N in RSA encryption) of at least 3072 bits (Smart et al., 2014).

5 Elliptic Curves Cryptography

For the scope of this project Elliptic Curves can be written in the form

$$y^2 = x^3 + ax + b$$

for Elliptic Curves Cryptography constants a and b must be in a form such that

$$4a^3 + 27b^3 = 0$$

A property of elliptic curves is that they are symmetric over the x-axis. This is because if there is a point (w, v) on an elliptic curve, then $(w, -v)$ is on the elliptic curve, because $(v)^2 = (-v)^2$. For elliptic curves, addition is defined in an unconventional way.

- For two points on an elliptic curve $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, a line is drawn through P and Q, where that line intersects the elliptic curve is our third point $R' = (x_3, y_3)$, then make $P + Q = R = (x_3, -y_3)$.

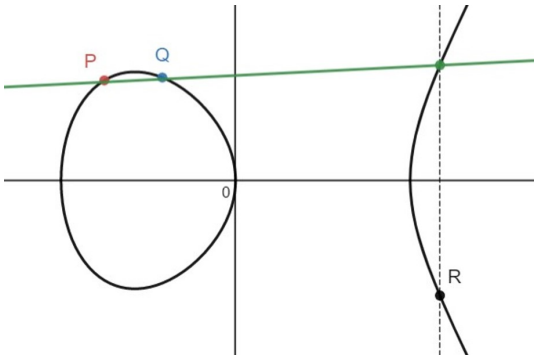

Figure 2: elliptic curve point addition

- If $P = (x_1, y_1)$ and $Q = (x_1, -y_1)$ then it is clear to see that the line going through P and Q does not touch the elliptic curve at a point R, so define $Q + P = O$ where O is the point at infinity.

- If P = Q a line drawn through P and tangential to the elliptic curve, the place where the line intersects the elliptic curve is $R' = (x_3, y_3)$. We say $P + Q = R = (x3, -y3)$

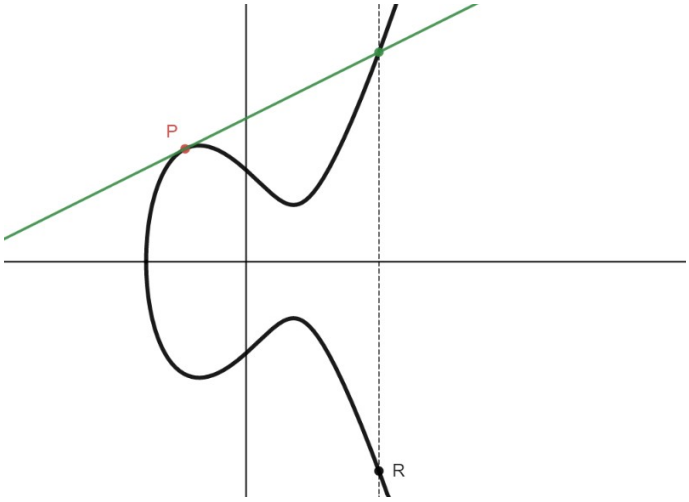

Figure 3: elliptic curve point doubling

This final step can be called point doubling where $P + P = 2P$ and

$$\underbrace{P + P \ldots + P}_{n\ times} = nP$$

Using these 3 operations it can be shown to create and abelian group. Elliptic curve addition can be explained in terms of simple operations.

To add two points

$$P = (x_1, y_1)\ and\ Q = (x_2, y_2)$$

set variable

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1}$$

and then

$$P + Q = (x_3, y_3) = (\lambda^2 - 2x_1, \lambda(x_1 - x_2) - y_1)$$

To double a point $P = (x, y)$

set variable

$$\lambda = \frac{3x_1^2 + a}{2y_1}$$

and then

$$2P = (\lambda^2 - 2x_1, \lambda(x_1 - x_2) - y_1)$$

For example, for points $P = (0, 3.162)$ and Q $= (-1.847, 0)$ over the curve elliptic curve

$$y^2 = x^3 + 2x + 10$$

then

$$\lambda = \frac{0 - 3.162}{-1.847 - 0} = 1.712$$

and

$$P + Q = (x_3, y_3) = (\lambda^2 - 0 + 1.847, -\lambda(x_3 - 0) - 3.162) = (4.777, -11.339)$$

To double the point P then

$$\lambda = \frac{0^2 + 2}{2(3.162)} = 0.3163$$

$$2P = (0.3163^2 - 2(0), -0.3163(x_3 - 0) - 3.162) = (0.1, -3.1936)$$

5.1 Elliptic Curves modulo p

This kind of encryption is mostly done modulo p where p is prime number.

This makes the "elliptic curves" in elliptic curve cryptography look less like an elliptic curve and more like a collection off random points.

The Elliptic Curve can be written in the form

$$y^2 mod\ p = x^3 + ax + b mod\ p$$

where x and y are points on the curve and b and c are constants.

The following needs to hold for encryption to work

$$(4a + 27b)\ mod\ p \neq 0\ mod\ p$$

To add two points

$$P = (x_1, y_1)\ and\ Q = (x_2, y_2)$$

set variable

$$\lambda = (y_2 - y_1)(x_2 - x_1)^{-1}$$

and then

$$P + Q = (x_3, y_3) = (\lambda^2 - 2x_1, \lambda(x_3 - x_1) - y_1)\ modulo\ P$$

To double a point $P = (x, y)$

set variable

$$\lambda = (3x_1^2 + a)(2y_1)^{-1}$$

and then calculate

$$2P = (\lambda^2 - 2x_1, \lambda(x_3 - x_1) - y_1) mod P$$

This slight change is used because doing this process modulo p makes calculating the public key from the private key easier lot easier then with normal elliptic curves and given a large enough p and well made elliptic curves in this form does not compromise the security. ECC is also proven to produce a finite abelian group modulo a prime number.

For a small example we can pick the curve

$$y^2\ mod\ 7 = x^3 + x + 4\ mod\ 7$$

This modulo elliptic curve leads to a group of 5.

+	0	(4,3)	(6,4)	(6,3)	(4,4)
0	0	(4,3)	(6,4)	(6,3)	(4,4)
(4,3)	(4,3)	(6,4)	(6,3)	(4,4)	0
(6,4)	(6,4)	(6,3)	(4,4)	0	(4,3)
(6,3)	(6,3)	(4,4)	0	(4,3)	(6,4)
(4,4)	(4,4)	0	(4,3)	(6,4)	(6,3)

5.2 Security of Elliptic Curve Cryptography

To create a public and Private key in an Elliptic curve cryptography system one must pick a point P on an elliptic curve. For a natural number n find nP. The key to the security of ECC systems is the difficulty of finding n given nP and P, so n is the private key, and nP and P are the public key. This all assumes a sufficiently large n and over a sufficiently large. The European Agency for Network and Security Information recommend a key length (the value of P in modulo ECC) of at least 256 bits.

EEC cryptography systems usually require less computational power and has a smaller private key compared to RSA encrypting. ECC is can be used as a digital signature algorithm.

6 Digital Signatures

When sending a message to someone it can be useful to apply a signature. A digital signature is a message or value that has been encrypted with a private key and which can be decrypted using the relevant public key. If an encrypted digital signature was written publicly it could be copied and stolen without posession of the private key. So before encrypting a plain text message with the intended recipient's public key the sender would add their encrypted message at the end. After the receiver receives the message and verifies and decrypts the message they will find the sender's original plain text and encrypted message. They can use the sender's public key to decrypt the signature and see a message that would be unique to the sender like their name.

This works because if the receiver assumes or knows that only the sender knows their private key, then even though they don't know how the message was encrypted they can decrypt it using the sender's public key.

One of the most common digital signature algorithms is the Elliptic Curve Digital Signature Algorithm (ECDSA) which utilises the basic concepts Laid out in the ECC section above.

7 Cryptographic hash functions

Cryptographic hash functions are unique functions.

They are characterised by

- Fixed length
- Deterministic - The same input always leads to the same output
- Slight changes to input significantly alter the output
- Fast to calculate
- One-way functions - It is nearly impossible to find the input if only given the output
- Collision-resistant - two input can't lead to the same output

Passwords for websites are a great example of hash functions in which every password is transformed into a fixed length hash after a hash function. Hash functions are designed to be one way functions in which it is much easier to find the output given then input then it is to find the input given the output.

7.1 Secure Hashing Algorithm - 256

One of the most common algorithms to mine Bitcoins is the Secure hashing algorithm - 256 (SHA - 256) hash function (Rhodes, 2020). To obtain the acceptable hash, miners need to play with the 'nonce' in an incremental way.

SHA-256 is part of the SHA-2 family which is a group of hashing algorthms. SHa - 2 family is commonly used in online security (Lake, 2022) and is the current hash function used to mine bitcoin.

Hash functions are desgined is such a way that small changes in an input lead to large changes in the output.

eg

Input: Brian Output: 5714e04739071aabdf34a209fcec4c33976a969dd2ca1c5007b406b2d8642bc5

Input: Bryan Output: 674c1f08fac053e604366eb24f2123568e367479301d4dd14e6109ca85abda1b

7.1.1 Shift Right

Shift Right is denoted by ShR(A,n) where A is the original message and n is the amount is it shifted by.
Just shifts stuff right and adds a zero on the left.
eg For ShR(111100001100, 4)
Original message: 111100001100
Shifted right 4 times: 000011110000

7.1.2 Rotate Right

Rotate Right is denoted by RotR(A,n) where A is the original message and n is the amount is it shifted by. Is similar to the right shift operation but instead of the right most characters being pushed of and adding zeros to the left.
Eg For RotR(111100001100, 4)
Original message: 111100001100
Rotated right 4 times: 110011110000

7.1.3 And and Or and Exclusive Or

The And and Or and Exclusive Or operations, denoted by $\wedge$, $\vee$ and $\oplus$ respectively, takes two binary messages and compares each digit.

In the And operation if there is only one 1 or no 1s between both messages in a given position then a 1 is put in the new message in the same position in the new message, if not then a zero is added.

Eg $11110000 \wedge 10101010 = 10100000$
Original message 1: 11110000
Original message 2: 10101010
New message: 10100000

In the Or operation if there is one 1 or two 1s between both messages in a given position then a 1 is put in the new message in the same position in the new message, if not then a zero is added.

Eg $11110000 \vee 10101010 = 01011010$
Original message 1: 11110000
Original message 2: 10101010
New message: 10100000

In the Exclusive Or operation if there is only one 1 between both messages in a given position then a one is put in the new message in the same position in the new message, if not then a zero is added

Eg $11110000 \oplus 10101010 = 01011010$
Original message 1: 11110000
Original message 2: 10101010
New message: 10100000

7.1.4 Binary Addition

Lastly binary addition is used mod 2^{32}. The modulo is so the messages are 32 numbers in length. It can be assume that all binary addition in this section is being done mod 2^{32}

7.1.5 Functions

There are a different function that are used in the SHA-256

$$Ch(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z)$$

$$Maj(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$$

$$\varepsilon_0(X) = RotR(X,2) \oplus RotR(X,13) \oplus RotR(X,22)$$

$$\varepsilon_1(X) = RotR(X,6) \oplus RotR(X,11) \oplus RotR(X,25)$$

$$\sigma_0(X) = RotR(X,7) \oplus RotR(X,18) \oplus ShR(X,3)$$

$$\sigma_1(X) = RotR(X,17) \oplus RotR(X,19) \oplus ShR(X,10)$$

7.1.6 Constants

There are many constants used in the hash function. They are derived from the cube root of the first 64 prime numbers and then converted into binary form. These constants are denoted by k_{0-63}

7.2 How SHA-256 Works

SHA-256 is designed so that it is impossible to to reverse engineer an input from an output.
When a message is inputted into the function is
It will be turned into bites based on the ascii table
Eg the string “ipek” Would turn into
i = 73 p = 112 e = 101 k = 107

And then would be turned into binary

73 = 1001001

112 = 1110000

101 = 1100101

107 = 1101011

And we put the together to make

1001001111000011001011101011

For a function to be used for the next steps we need to pad it out so it is a 512 length binary code. This is done by using padding.

Padding is performed by adding a one at the end of our binary message and then adding zeros until the message is 448 bits long so this message becomes

1001001111000011001011101011100000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000

The last 64 characters are used to show the length of our original message, the original message has a length of 28. In binary this is 11100. Zeros are filled until it is of length 64.

0000000000000000000000000000000000000000000000000000000000011100

And we add it to the end of the message.

1001001111000011001011101011100000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000

00000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000011100

This length 512 binary string is turned into 8 64-long words. In our example our words would be

$W_0 = 10010011110000110010111010111000$
$W_1 = 00000000000000000000000000000000$
$W_2 = 00000000000000000000000000000000$
$W_3 = 00000000000000000000000000000000$
$W_4 = 00000000000000000000000000000000$
$W_5 = 00000000000000000000000000000000$
$W_6 = 00000000000000000000000000000000$
$W_7 = 00000000000000000000000000000000$
$W_8 = 00000000000000000000000000000000$
$W_9 = 00000000000000000000000000000000$
$W_{10} = 00000000000000000000000000000000$
$W_{11} = 00000000000000000000000000000000$
$W_{12} = 00000000000000000000000000000000$
$W_{13} = 00000000000000000000000000000000$
$W_{14} = 00000000000000000000000000000000$
$W_{15} = 00000000000000000000000000011100$

Out of the 16 words we derive 48 more from them using the equation

$$W_n = \sigma_1(W_{n-2}) + W_{n-7} + \sigma_0(W_{n-15}) + W_{n-16}$$

To make the 17th word say

$$W_{16} = \sigma_1(W_{14}) + W_9 + \sigma_0(W_1) + W_0$$

This process is continued until W_n for $n = 16, 17 \ldots 64$ is derived.

7.2.1 More Constants

In the SHA-256 there are different starting constants. They are derived from the square roots of the first 8 prime numbers and then remove the part in front of the decimal point. These are called hash values

$a = \sqrt{2} \Rightarrow 0.4142135624 \times 2^{32} = 01101010000010011110011001100111$
$b = \sqrt{3} \Rightarrow 0.7320508075 \times 2^{32} = 10111011011001111010111010000101$
$c = \sqrt{5} \Rightarrow 0.2360679775 \times 2^{32} = 00111100011011101111001101110010$
$d = \sqrt{7} \Rightarrow 0.6457513111 \times 2^{32} = 10100101010011111111010100111010$
$e = \sqrt{11} \Rightarrow 0.3166247904 \times 2^{32} = 01010001000011100101001001111111$
$f = \sqrt{13} \Rightarrow 0.6055512755 \times 2^{32} = 10011011000001010110100010001100$
$g = \sqrt{17} \Rightarrow 0.1231056256 \times 2^{32} = 00011111100000111101100110101011$
$h = \sqrt{19} \Rightarrow 0.3588989435 \times 2^{32} = 01011011111000001100110100011001$

These constants to start the compression function.

7.2.2 Compression Function

2 new values are created by doing the following operation .

$$T_1 = \varepsilon_1(e) + Ch(e, f, g) + h + K0 + W0$$

$$T_2 = \varepsilon_0(a) + Maj(a, b, c)$$

Next the constants a-h are shifted so the value of every letter changes to the one above it and a becomes $T_1 + T_2$. Except for e witch becomes $d + T_1$

$a = T1 + T2$
$b = a$
...
$e = d + T1$
...
$h = g$

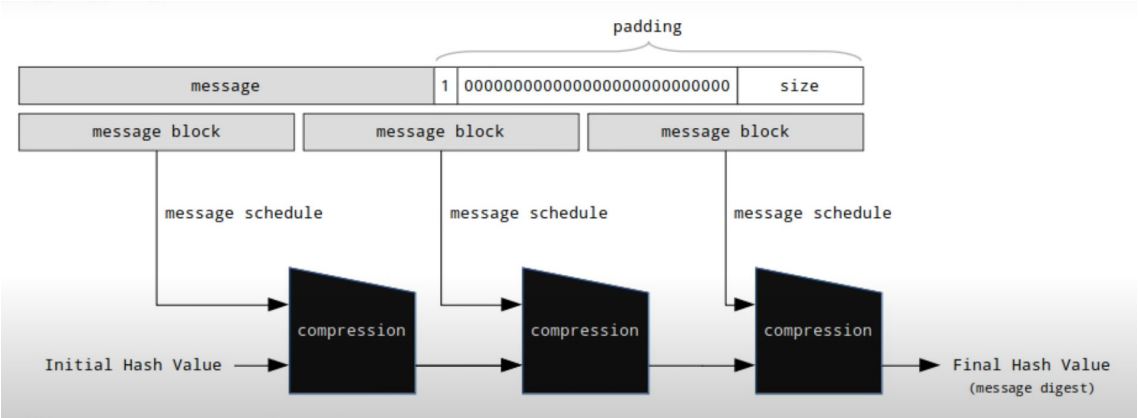

Figure 4: Compression function

This process is repeated with our new variables and with W_1 and K_1 instead of W_0 and K_0.

This process is repeated 64 times with each different word.

Once the new hash values are acquired they are added to the original hash values

$old\ a\ +\ new\ a\ =\ new\ new\ a$

...

$old\ h\ +\ new\ h\ =\ new\ new\ h$

These new values of a to h would be called the new hash values. And this process would be the first round of compression. So the new values of a to h would be values after the first compression

After the process of compression is completed 3 times with the hash values being updated each time, we get our final hash values.

7.2.3 Changing Form

After the final hash values are found they are converted from binary into hexadecimal form. The end result is the final output/message digest.

8 Cryptocurrencies

The central entities in cryptocurrency blockchains are blocks, transactions, inputs, and outputs. A blockchain is a distributed database or ledger that is shared among the nodes of a computer network. The goal of blockchain is to allow digital information to be recorded and distributed, but not edited. In this way, a blockchain is the foundation for immutable ledgers, or records of transactions that cannot be altered, deleted, or destroyed. This is why blockchains are also known as a distributed ledger technology (DLT). Each block in the chain is given an exact timestamp when it is chained to the previous block, which means the data is chained together in chronological order.

The core building blocks for a blockchain framework are

- A shared ledger - In which changes are only made when 51% of the nodes agree on a modification.
- Cryptography - that ensures transparency, transactions are secure authenticated and verifiable.
- Trust Systems and Consensus Transactions - that must be endorsed by the relevant participants.
- Business rules and smart contracts - Rules based on collective agreement much be in place (Baset et al., 2019, p. 10).

An important feature of a cryptocurrency blockchain data model is the lack of a central location to store the balance of an account. Thus, account balance must be calculated as unspent coins (outputs). Another feature is that coins (outputs) can only be spent fully. This means that in order to transfer only half the value present at an address, the transaction will send half of the value from an account A to account B, and send the other half of the value to account C (instead of A). The procedure makes it impossible to distinguish whether the second half was given back as change to the original owner or sent to a completely separate account.

Many companies now accept cryptocurrencies as payment, for example Microsoft and AT&T (Beigel, 2023).

A big problem with cryptocurrencies is that they can only handle a certain amount of transactions in a given day. For bitcoin it is 300,000 transactions in a day and for Ethereum it is 1.4 mil transactions a day.

8.1 Bitcoin

In 2009 a digital currency, Bitcoin, was introduced by the developer Satoshi Nakamoto. It was the first cryptocurrency that enables peer-to-peer transactions without involving any intermediators eg banks (Nakamoto, 2008).

A blockchain consists of blocks, blocks contain transactions, and each block contains information about the unique hash identifier of a block and a time of the block in Unix epoch format[11]. A Unix epoch is a time system describing the number of seconds from 00:00:00 UTC on January 1, 1970 (also known as the Unix epoch). 1270916552 (for 2010-04-10 19:22:32) would be an example of a block timestamp.

Each transaction has a unique hash identifier that refers to a specific block; therefore, the exact timestamp of a transaction is inherited from and identical to a block timestamp. The inputs and outputs contain detailed information on transactions—the amount of the sender(s) and receiver(s). The transaction value for the Bitcoin blockchain is expressed as 100,000,000 Satoshis for every 1 Bitcoin.

Ownership of Bitcoin can be proven mathematically through public-key cryptography. However, cryptography alone cannot guarantee that one particular coin hadn't previously been sent to someone else. The double-spend problem refers to the issue of needing to find consensus on a history of transactions.

In the case of Bitcoin mining, a succession of blocks can be mathematically proven to have been stacked in the correct order with a certain commitment of resources. The process hinges on the mathematical properties of a cryptographic hash — a way to encode data in a standardized manner. Hashes are a one-way encryption tool, meaning that decrypting them to their input data is nearly impossible, unless every possible combination is tested until the result matches the given hash.

This means that altering even the tiniest component of a block would noticeably change its

expected hash — and that of every following block, too. Nodes would instantly reject this incorrect version of the blockchain, protecting the network from tampering. Nodes accept the version of the blockchain that is present in the majority (ie over 51%) and rejects any versions present in the minority. Through the difficulty requirement, the system guarantees that Bitcoin miners put in real work — the time and electricity spent in hashing through the possible combinations. This is why Bitcoin's consensus protocol is called "proof-of-work," to distinguish it from other types of block-creation mechanisms. But, how long does it take to mine 1 Bitcoin. One BTC typically takes around 10 minutes to create, although this is only true for strong processors. The Bitcoin mining hardware you use will determine how quickly you can mine.

Data Examples I will present data examples to demonstrate how each is represented in the block chain. (Liiv, 2021, p. 8)

• A block that has two transactions —one row in the blocks table and two rows in the transactions table, linked using the unique identifier of a block.

Block number 78400 has 2 transactions.

Blocks:

Id	hash	time
78400	000000000068c8c17fdd4e9ceec82a874da300601619a0495ba9117b4267a313	1283795840

Transactions:

Id	hash	blockID
111720	e16f4fc80f97b6e59e5077c00819775a335bdc946d33d66a93f7c5aca3bbfd9a	78400
111721	7c1ede95cfc8f536b7d75483de04c30fc20278825be37296d81d43a5f70a0299	78400

This is an example of a block that has two transactions.

- A block reward transaction (new coins were generated and awarded by the blockchain system to cryptocurrency miners, who are responsible for securing the blockchain and processing transactions) -one row in the blocks table, one row in the transactions table (a block reward is often, but not always, the first transaction in a block), one row in the outputs table and no rows in the inputs table (since there is no source address for a newly generated coin. The reward is received by a destination address, linked using the

unique identifier of a transaction, which in turn is linked using the unique identifier of a block.

Block reward was sent to an address: 1JBSCVF6VM6QjFZyTnbpLjoCJTQEqVbepG Amount: 50

Blocks:

Id	hash	time
79880	00000000002e872c6fbbcf39c93ef0d89e33484ebf457f6829cbf4b561f3af5a	1284561413

Transactions:

Id	hash	blockID
114894	f5d8ee39a430901c91a5917b9f2dc19d6d1a0e9cea205b009ca73dd04470b9a6	79880

Outputs:

Id	dstAddress	value	txID	offset
135765	1JBSCVF6VM6QjFZyTnbpLjoCJTQEqVbepG	5000000000	114894	0

This is an example of a block reward transaction (new coins were generated)

8.2 Public vs Private Keys in Cyrptocurrencies

The public key in a cryptocurrency is known by everyone, it is the code that people use to send bitcoin and a public facing code used when sending a currency.

The private key is private and is used like a normal password for someone's cryptoaccount. There is no way to recover one's cryptoaccount if the private key is lost. Many exchanges will hold a person's private key meaning that the currency technically belongs to them.

The public key is derived from the private key but is designed so that the private key can't be derived from the public key.

Similarly to RSA a transaction from Alice to Bob can be done she can encrypt the transaction simply by using bobs public key, but then Bob can only decrypt the message using his private key.

8.3 Transaction

When Alice sends transactions to Bob, Alice publicly signs the transaction using her private key. This public signature is then broadcasted out to all the nodes/miners who work to verify the transaction

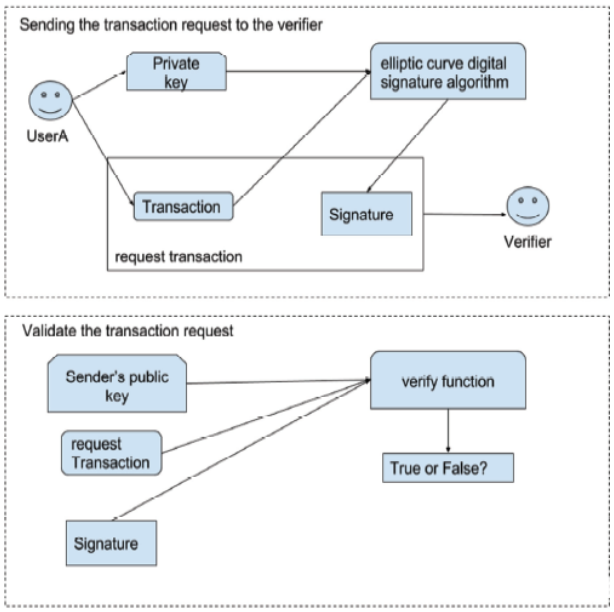

Figure 5: Cryptocurrency Transaction

8.4 Proof of Work

Producing a Proof of Work (PoW) is a random process with a low probability of success. There is a large amount of trial and error before a valid proof of work is generated. PoW schemes involve solving a mathematical puzzle in which the solution is easily verified. Puzzles require a lot of energy of solve on a large scale, this is why miners that find the solutions are rewarded.

8.5 Mining

Mining is the process of verifying and approving transactions usually using high performance mining computers. The miners create valid blocks to add transaction records to the public ledger of a blockchain (LO Swee Won, 2021)(pg 89).

Before a block is validated it needs to be confirmed that the sender has the funds to do so, this is checked by going through the senders transaction history on the blockchain. The block

also needs to be signed by the senders digital signature.

To have a block truly validated it needs to be added to the block chain. It would make sense for the miners broadcast the new block to the other miners in the block chain, and after each miner receives the new block they will add it to the blockchain. The problem is that this could cause confusion when each node is simultaneously trying to transmit it's own blocks.

To make adding blocks to the block chain more secure and safe they are added by using a consensus algorithm which must be agreed upon by all the miners.

Finding a appropriate hash to fulfil a certain difficulty is very hard and requires a lot of computational power and energy. In proof of work systems finding difficulty is usually a random process with low probability of success. To incentivise miners to keep mining they are rewarded with the native cryptocurrency of the blockchain they are mining. Miners are also compensated by transaction fees that go along with many cryptocurrencies now a days including bitcoin.

With ever increasing innovation in computers, miners are encouraged to invest in more hardware the make mining more efficient. This makes it so the difficulty is constantly getting harder (Sharma, 2022).

Before mining, one need to consider

- Must examine the amount of difficulty specific to the cryptocurrency they wish to mine
- The future value of the cryptocurrency
- The cost of hardware and cost of electricity
- The value of the transaction fees

Miners often build or join mining pools by combining their efforts with other miners. Groups of miners who work together have a higher chance of earning rewards and splitting the profits (Lin William Cong, 2019). A larger pool offers higher risk-sharing benefits. In addition, members of a mining pool pay a fee to be a part of the pool.

8.6 Bitcoin miners

Bitcoin miners cycle through trillions of hashes every second until they find one that satisfies a condition called "difficulty." Both the difficulty and the hash are very large numbers expressed in bits, so the condition simply requires the hash to be lower than the difficulty.

Difficulty readjusts every 2016 Bitcoin blocks — or approximately two weeks — to maintain a constant block time, which refers to how long it takes to find each new block while mining.

This is done by finding a hash value that starts with a certain amount of zeros. After an appropriate hash value is found an honest miner will attach it to a valid transaction proposal creating a block. The valid hash value is attached to the block.

The hash generated by miners is used as an identifier for any particular block and is composed of the data found in the block header. The most important components of the hash are the Merkle root — another aggregated hash that encapsulates the signatures of all transactions in that block — and the previous block's unique hash.

The creator Satoshi Nakamoto set a schedule so the rewards continue to decrease as the number of bitcoins mined increases. After each 210000 blocks are mined the compensation for each block mined is cut in half starting at 50BTC for each block (Draupnir, 2016).

Crypto mining (in Bitcoin's case) is a computer operation that creates new Bitcoin and tracks transactions and ownership of the cryptocurrency. Bitcoin mining is energy-intensive and can produce significant financial rewards.

Miners need to determine whether they are willing to invest the required initial capital in hardware and they also need to determine the future value of Bitcoin and the level of difficulty before committing to it.

When both Bitcoin prices and mining difficulty fall, it usually means fewer miners are mining BTC and that acquiring BTC is easier. Nonetheless, expect more miners to compete for fewer BTC as Bitcoin prices and mining difficulty climb.Miners are constantly validating and then adding more transactions to the blockchain but transactions cannot be removed from the centeral ledger so the blockchain can only get bigger

8.7 51% problem

With a decentralized network which is kept secure through different people there is a concern that is could be possible for one person to change a forge the blockchain if they had enough nodes to be able to manipulate the blockchain.

- In a well established network larger numbers of people will participate, connect a very significant number of nodes and so it will take a large amount of initial investment, with little payoff because value of currency will drop.
- In BTC history this was close to happening with a pool of miners. To avoid this lots of the miners left the pool.
- For small networks it is not hard to muster 51% of the computing power but there is very little financial incentives (Baset et al., 2019, p. 426).

8.8 Proof of Stake

Proof of Stake (PoS) is the alternative method to PoW for verifying transitions. It is a more popular method with new cryptocurrencies(cite this) and as of September 2022 used by Ethereum the second most cryptocurrency blockchain (Holmes, 2022). In PoW, miners in a given blockchain have an interest in the integrity of the blockchain because of the capital they have spend. In a PoS system a miner creates a smart contract on the blockchain in which they front capital in the form of the native currency of the blockchain. The capital fronted is used as collateral in case of the miner acting dishonestly or lazliy. The miner is then responsible for checking that blocks appended to the blockchain are valid and to ocationally create new blocks themselves(OG cite bad find better). The amount of stake that is posted by a miner is proportional to the chance of mining the next block. In very simple forms if a miner own 10% of the entire stake they have a 10% chance of mining the next block.

In a PoS model the main problem is what is called the "nothing at stake model". When 2 miners create a block at the exact same time it leads to two versions of the block chain.

This is not a problem for PoW systems. This is because in a PoW system miners must decide which version of the blockchain they want to mine of top of, since it would require much more

computing power to mine on both versions of the blockchain (LO Swee Won, 2021)[p. 107].

PoS miners do not need to choose a side that they wish to mine because there is very little computational power needed to add to each side.

8.9 Differences between PoW and PoS

PoS is more common for new emerging cryptocurrencies in contrast to PoW because it

- Less energy intensive
- Has lower barriers to entry for miners
- Reduced hardware requirements
- Since it is easier to to join one of the nodes and start mining it leads to more nodes being created.

9 Conclusion

Mathematics plays a critical role in modern cryptography techniques and the development of cryptocurrencies. The mathematical concepts used in cryptography provide a secure foundation for transactions and data transmission, while mathematical algorithms underpin the mining and exchange of cryptocurrencies.

Cryptography is the science of secure communication in the presence of third-party adversaries. It involves the use of mathematical concepts such as number theory, algebra and probability theory to develop secure encryption and decryption protocols (Goldwasser & Micali, 1983). Cryptography plays a crucial role in ensuring the security and integrity of data transmission, including online transactions, communication, and storage. For example, public-key encryption algorithms such as RSA and elliptic curve cryptography (EEC) use advanced mathematical concepts to encrypt and decrypt sensitive data (Alfred J. Menezes & Vanstone, 1996).

Cryptocurrency, on the other hand, uses mathematical algorithms such as hashing, digital signatures, and proof of work to ensure the secure exchange and storage of digital assets. The

blockchain technology that powers cryptocurrencies is built upon complex mathematical algorithms that provide a decentralized, tamper-proof network. These algorithms facilitate the mining of cryptocurrencies such as Bitcoin and support the secure transfer of digital assets (Nakamoto, 2008).

Mathematics is an essential component of modern cryptography techniques and the development of cryptocurrencies. The secure encryption and transmission of data, as well as the secure exchange and storage of digital assets, relies on advanced mathematical concepts and algorithms. Therefore, a solid understanding of mathematics is critical for the successful development and deployment of secure communication and digital transaction protocols.

References

Alfred J. Menezes, P. C. v. O., & Vanstone, S. A. (1996). *Handbook on applied cryptography* (Vol. 5). CRC press.

Baset, S. A., Desrosiers, L., Gaur, N., Novotny, P., O'Dowd, A., Ramakrishna, V., Sun, W., & Wu, X. ((2019). *Blockchain development with hyperledger* (Vol. 1). Packt.

Beigel, O. (2023). Who accepts bitcoin as payment? *99bitcoins*. https://99bitcoins.com/bitcoin/who-accepts/

Criddle, C. (2021). Bitcoin consumes 'more electricity than argentina'. *British Broadcasting Channel news*. https://www.bbc.com/news/technology-56012952

Damico, T. M. (2009). A brief history of cryptography. *inquiries journal*, *1*. https://www.inquiriesjournal.com/articles/1698/a-brief-history-of-cryptography

Draupnir, M. (2016). What is the bitcoin mining block reward? *Bitcoin Mining*. https://www.bitcoinmining.com/what-is-the-bitcoin-block-reward/

Goldwasser, S., & Micali, S. (1983). Probabilistic encryption. *Journal of computer and system science*. https://doi.org/28,270-299

Holmes, F. (2022). Decision to switch ethereum to proof-of-stake may have been based on misleading energy fud. *Forbes*. https://www.forbes.com/sites/greatspeculations/2022/09/21/decision-to-switch-ethereum-to-proof-of-stake-may-have-been-based-on-misleading-energy-fud/?sh=3e522fcd384a

Lake, J. (2022). What is sha-2 and how does it work? *Comparitech*. https://www.comparitech.com/blog/information-security/what-is-sha-2-how-does-it-work/

Liiv, I. (2021). *Data science techniques for cryptocurrencies blockchains* (Vol. 9). Springer.

Lin William Cong, J. L., Zhiguo He. (2019). Decentralized mining in centralized pools. *NATIONAL BUREAU OF ECONOMIC RESEARCH*. https://www.nber.org/system/files/working_papers/w25592/w25592.pdf

LO Swee Won, D. L. K. C., WANG Yu. (2021). *Blockchain and smart contracts* (Vol. 4). Packt.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Rhodes, D. (2020). Sha-256 cryptographic hash algorithm. *Komodo*. https://komodoplatform.com/en/academy/sha-256-algorithm/

Sharma, T. K. (2022). What is proof-of-work? *Blockchain Council*. https://www.blockchain-council.org/blockchain/what-is-proof-of-work/

Smart, N. P. (2016). *Cryptography made simple* (Vol. 1). Springer.

Smart, N. P., Rijmen, V., Gierlichs, B., Paterson, K. G., Stam, M., Warinschi, B., & Watson, G. (2014). Algorithms, key size and parameters report – 2014. *European Union Agency for Network and Information Security.*

Stalling, W. (2021). *Cryptography and network security: Principles and practice* (Vol. 5). Pearson.

Vaudenay, S. (2005). *Public key cryptography* (Vol. 8). Springer.