AN ABSTRACT OF THE THESIS OF

Sophia Crossen for the Master of Science

in Mathematics presented on April 23, 2015.

Title: The Mathematics of Bitcoin

Abstract approved:

Bitcoin is among the first successful, fully implemented cryptocurrencies. As it slowly emerges into mainstream use, it is necessary to understand how a transaction works, involves an entirely digital currency. This paper provides an overview of a bitcoin transaction from the point of view of the user, looking at the details and security of an individual transaction, transaction blocks, and the Bitcoin public ledger, or block chain. It also discusses the mathematics implemented to secure and maintain the trust in the Bitcoin network. Security for the individual transaction is achieved through use of digital keys and digital signatures. Elliptic Curve Cryptography using the secp256k1 curve and Elliptic Curve Digital Signature Algorithm are the algorithms most commonly used in a bitcoin transaction. The Secure Hash Algorithm 256 (SHA256) condenses information in a one-way hash function. Double SHA256 insures the security of individual transactions and secures the block chain against tampering.

Keywords: bitcoin, transaction block, block chain, SHA256, Merkle tree, Merkle root, elliptic curve, secp256k1, digital key, pubkey script, signature script, block header, nonce, target difficulty

THE MATHEMATICS OF BITCOIN

A Thesis

Presented to

The Department of Mathematics

EMPORIA STATE UNIVERSITY

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

Sophia Lillian Muse Crossen

May 2015

Approved by the Department Chair

Approved by the Dean of the Graduate School and Distance Education

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 2

CHAPTER 4

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 3

1 INTRODUCTION

Cryptocurrency, a relatively new form of currency, uses cryptography, networking and open source software to establish trusted transactions and control creation of new units (Greenberg 2011). In his article, "bmoney," Wei Dai describes cryptocurrency in rather broad terms, never actually naming his proposed medium of exchange (1998). Dai described two protocols to define an electronic medium of exchange and a method of enforcing a contract (1998). Dai's premise inspired much speculation over the next decade. In 2008, the white paper entitled "Bitcoin: A Peer-to-Peer Electronic Cash System," was published by an anonymous author under the pseudonym Satoshi Nakamoto, inspiring the creation of many cryptocurrencies. Nakamoto proposed "an electronic payment system based on cryptographic proof instead of trust, allowing any two willing parties to transact directly with each other without the need of a trusted third party" (2008, p. 1). He goes on to outline a basic system for establishing a cryptocurrency (Nakamoto 2008). In 2009, bitcoin was introduced (Velde 2013). A trusted third party, like a bank, is no longer required for processing electronic payments (Nakamoto 2008).

In a traditional bank account, all account information and transaction details are kept private with none of the details available to anyone not a party to the interaction (Fig. 1.1). In a bitcoin transaction, the transaction is entirely open to the public except the

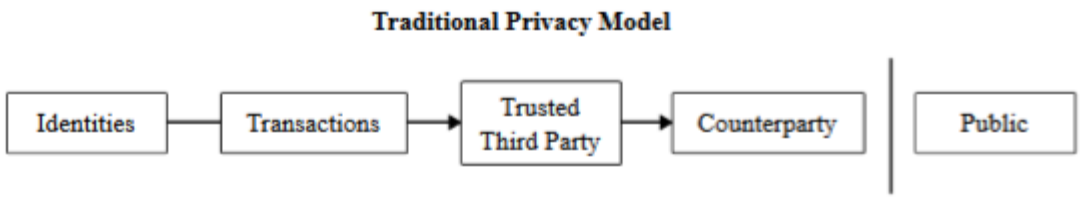

Figure 1.1 (Nakamoto 2008)

identities of the parties involved (Fig. 1.2) (Nakamoto 2008). While it is often possible through much research to determine the identities of the parties involved, these identities are not kept as part of the transaction record, allowing for pseudo anonymity (Bryans

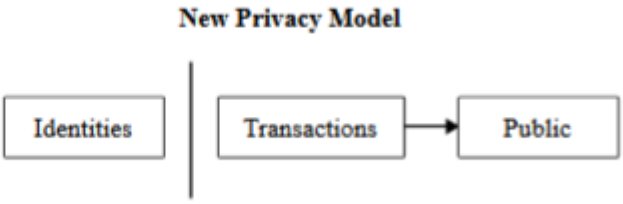

Figure 1.2 (Nakamoto 2008)

2014). The entire transaction becomes part of the public ledger, known as the *block chain*. In this ledger, each transaction is a line in the ledger, and each bitcoin "account" is a different page of the ledger. Because there is no central authority, in order to ensure the integrity of the public ledger, high levels of digital security are employed at multiple stages of the transaction, including digital keys and signatures and double SHA256 cryptographic hashes (Antonopoulos 2014). Understanding these functions and their implementation in a bitcoin transaction begins by exploring the parts of a transaction.

2 ANATOMY OF A TRANSACTION

Trade in bitcoin is now as easy as signing up with an online wallet service since, like any other currency, bitcoin is "stored" in a *wallet*. This wallet service, and other sites like it, may provide other services associated with trade in bitcoin, and are often *nodes* in the peer-to-peer Bitcoin network themselves. A bitcoin wallet is an electronic record of all bitcoin a user owns, or with which her digital signature is currently associated. Accessing an online bitcoin wallet is similar to accessing an online bank account. Users simply log into their account, or wallet, and either transfers money to another user's account or accepts transfers from other users (Antonopoulos 2014).

For an illustration of a transaction from the point of view of a typical bitcoin user, consider the following example transaction. Alice would like to buy an e-book from Bob, but the only form of payment Bob accepts is bitcoin (BTC). Alice doesn't have any bitcoin; she only has hard currency, specifically US dollars ($). First, Alice creates an account with a wallet service, making certain to choose one that provides currency conversion into bitcoin. Alice needs 0.01 BTC for her purchase from Bob. Her wallet provider tells her the current rate for 1 bitcoin in US dollars: $250 per bitcoin (although close to the current exchange rate, this amount is used as example only). Bob is asking $2.50 for his e-book. Alice decides to convert $10 into bitcoin using a debit or credit card, giving her a balance in her wallet of 0.04 BTC. Now, Alice can make her purchase. Still accessing her new wallet, she chooses the option to send bitcoin. Alice enters either Bob's wallet address or his email address into the wallet software and the amount she wishes to send to him, 0.01 BTC. Bob acknowledges the transaction, which is then broadcast across the Bitcoin network, and Alice has a 0.03 BTC balance in her wallet. As

this transaction is very small, the transaction can be considered confirmed immediately by Bob and Alice (Antonopoulos 2014), although validation of the transaction will take approximately 10 minutes and confirmation will take up to an hour. If she is at Bob's physical location and has a smart phone, she can instead use a wallet application to scan a QR code for Bob's wallet address ("Bitcoin Developer Guide" n.d.).

As an alternative to an online wallet, Bitcoin Core code may be installed on a user's computer and used to set up a private wallet. The user will need to have some basic programming knowledge as this method is not as friendly as the online wallets are made to be; i.e., no graphical user interface. This wallet acts in much the same way as the online wallets, allowing the user to request a new wallet address, send and receive bitcoin, record all bitcoin addresses currently owned by them, and submit transactions to the Bitcoin network for confirmation and inclusion in the block chain. By using this method, a user is able to create transactions while offline to be uploaded the next time they connect to the Internet ("Bitcoin Developer Guide" n.d.).

Each bitcoin and each wallet has an address. A user may request a new wallet address, but each bitcoin address is a unique identifier which reflects the history of that bitcoin and can be used to track that bitcoin back to the *genesis block* or *coinbase transaction* from which it originated. Each bitcoin address can represent any actual amount of bitcoin down to a single *satoshi* (the smallest allowed subdivision of a bitcoin), or 0.00000001 BTC, or as large as the total amount of bitcoin in existence. As bitcoin are pooled or divided in a transaction, or simply spent as is, their address will change to reflect their new histories (Antonopoulos 2014).

When Alice purchased her bitcoin, she paid in the traditional manner with an account in a traditional financial institution. Other options exist for acquiring bitcoin that provide a higher possibility of remaining anonymous, like selling goods or services online or making a purchase with cash at a local vendor (Antonopoulos 2014). For Alice, her provider then initiated a transaction. They chose the address of an output of 0.04 BTC from a previous transaction to be the input for this new transaction with Alice. If such an output does not exist, they create either an aggregating transaction (Fig. 2.1), combining smaller outputs into an appropriately sized input, or a distributing transaction (Fig. 2.2),

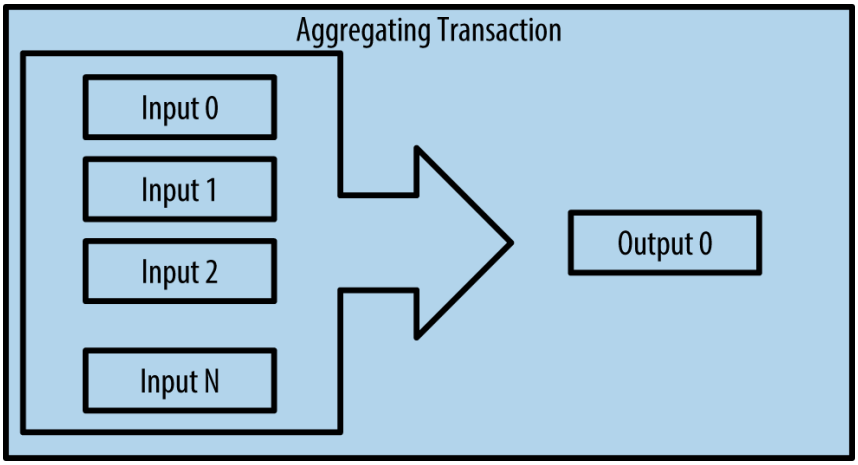

Figure 2.1 (Antonopoulos 2014)

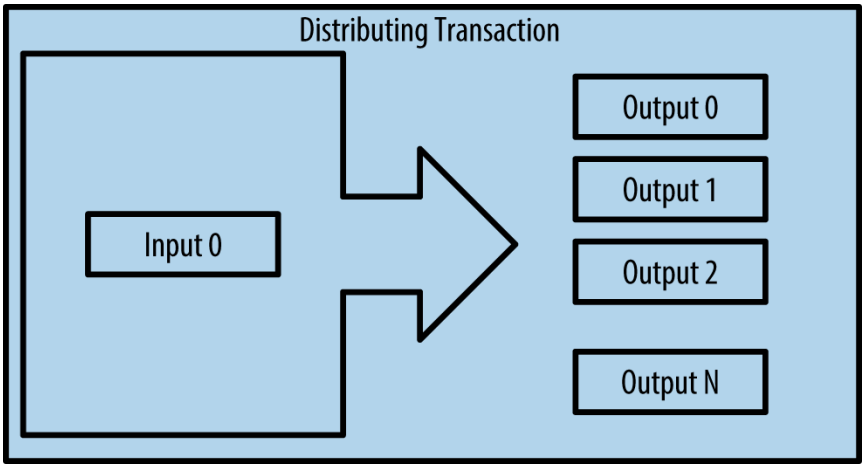

Figure 2.2 (Antonopoulos 2014)

dividing a larger input (Antonopoulos 2014). Her wallet provider's signature is then applied to this input, releasing it to a new owner, Alice, as an output in this transaction assigned to Alice's wallet address and encumbered with her public key hash or wallet address. In order for this bitcoin to be spent, the terms of the encumbrance must be met. In other words, Alice must apply her signature when including this as an input in a new transaction (Antonopoulos 2014).

Alice still needs to pay Bob. Alice includes the 0.04 BTC output as the input for her transaction with Bob. Remember, though, that Bob only needs 0.01 BTC. Alice doesn't want to pay him more than he's owed. She needs change back. The key to receiving change is breaking up the bitcoin into the desired amounts by designating output amounts and appropriate recipients (Fig. 2.3). In this example, Alice will designate her first output, #0, as the 0.01 BTC she is transferring to Bob's wallet. In a bitcoin transaction, any leftover amount from the input not specifically assigned to an output is considered a transaction fee, or payment, to the *bitcoin miners* who first confirm the transaction is valid (Antonopoulos 2014). So, if Alice stops here and sends the transaction for confirmation, she pays the remaining 0.03 BTC as a transaction fee. If she

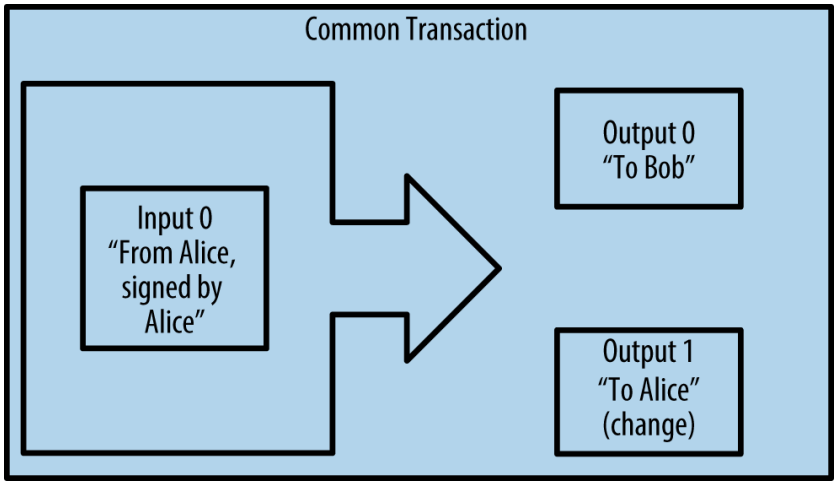

Figure 2.2 (Antonopoulos 2014)

doesn't want to lose this bitcoin, she has to include a second output, #1, in her transaction with Bob to assign the remaining 0.03 BTC back to her wallet. Most online wallets will automatically make this adjustment and charge any applicable transaction fees, making it transparent to the user. Now, Alice's wallet contains the address of the remaining 0.03 BTC as an unspent output and keeps this output recorded as available to be used as an input in a future transaction ("Bitcoin Developer Guide n.d.).

The *transaction*, including a record of all inputs and outputs, is now broadcast across the Bitcoin network to be added to a transaction block. Each transaction is hashed using a SHA256 hash function to create its unique transaction identifier to ensure that all data in the transaction cannot be altered without altering the transaction identifier (Table 2.1) (Antonopoulos 2014).

Transaction	
5bc1e89ba2b82f770da019bf147636361f83f2620b8cf2e3e47ce7f0255e4cfc	
Inputs	
113Uxv3jxJtduRGAAEuVMRpKcPuUKXDTGH	
Outputs	
1GSTmnp3JSCxaUymWgTF4eUhz7ui6HxhCz 0.07344311 BTC 113Uxv3jxJtduRGAAEuVMRpKcPuUKXDTGH 17.99849813 BTC	
Summary	
Size	225 (bytes)
Received Time	4/3/2015 17:00
Included In Blocks	350560 (2015-04-03 17:04:11 + 4 minutes)
Confirmations	1607 Confirmations
Inputs and Outputs	
Total Input	18.07294124 BTC
Total Output	18.07194124 BTC
Fees	0.001 BTC
Estimated BTC Transacted	0.07344311 BTC

Table 2.1 Transaction Included in Block #350650 ("Block 350650" n.d.)

Every *node*, or cooperative group of bitcoin miners, that receives this broadcast first confirms the validity of the transaction. Reasons for rejecting a transaction include: the transaction's syntax and data structure is incorrect, either the list of inputs or the list of outputs is blank, transaction size in bytes is too small, for each input the referenced output doesn't exist or has already been spent, the sum of the inputs is not greater than or equal to the sum of the outputs, etc. (Antonopoulos 2014). The node then adds the transaction to a pool of unverified transactions and rebroadcasts said transaction across the network to be certain all nodes have received it. Bitcoin miners select transactions from this transaction pool and, along with a *coinbase transaction*, build a new *transaction block*. Transactions are added to new blocks on a priority basis (Antonopoulos 2014).

Many factors are considered when deciding which transactions have a higher priority than others. These factors include, but are not limited to, the amount of the transaction fee being offered, how long it's been since these particular bitcoin have been spent, and available space in the transaction block. A new block is initiated as soon as the last block's proof-of-work is received and verified by the network, approximately every 10 minutes (Antonopoulos 2014).

Transaction block #350560 is shown below (Table 2.2, 2.3). The *block number* is the height of the block, or the number of blocks since block 0, the genesis block. Block number is one way to identify a transaction block, but is not necessarily unique. If there has been a *fork,* or alternative path, in the block chain, then the block number could refer to multiple transaction blocks. In Table 2.2 the block is not part of a fork since the height of this block is shown to be from the "Main Chain." (Antonopoulos 2014)

Summary	
Number Of Transactions	1661
Output Total	11,588.67136749 BTC
Estimated Transaction Volume	2,306.99196516 BTC
Transaction Fees	0.28227533 BTC
Height	350560 (Main Chain)
Timestamp	2015-04-03 17:04:11
Received Time	2015-04-03 17:04:11
Relayed By	F2Pool
Difficulty	46,717,549,644.71
Bits	404195570
Size	976.4306640625 KB
Version	2
Nonce	1217706329
Block Reward	25 BTC

Table 2.2 Block #350560 (“Block 350650” n.d.)

Hash 00000000000000000bb68584f973318a292d2ee4958c61b206e3ad73d5b1fc11
Previous Block 00000000000000000a2c1d40b71f295ece78f4100ab634173b5e88c7247cb33a
Next Block(s) 000000000000000001450e1fcca7d74877361c1e154ee0e888c211a212b4f4382
Merkle Root 159ceb3d0acd646b455fd6f0d5f5bc0258bee741c29119efd4aa000940ce6524

Table 2.3 Block #350650 Hashes (“Block 350650” n.d.)

The number of transactions includes the *coinbase transaction* (Table 2.4), a kind of reward to the bitcoin miner who first validates a new block. The number of transactions can be as few as one, just the coinbase transaction, or as many as will fill 250,000 bytes. An average transaction size is approximately 250 bytes (Antonopoulos 2014). The coinbase transaction is the primary method of payment for bitcoin miners and is the only way new bitcoin is created. Coinbase transactions require 100 confirmations

Transaction ID 12bbe05ed70c4664bf88a8294f0c37c9004524b83580bddb91b840341a06133f	
2015-04-03 17:04:11	
No Inputs (Newly Generated Coins)	1KFHE7w8BhaENAswwryaoccDb6qcT6DbYY 25.28227533 BTC

Table 2.4 Coinbase Transaction ("Block 350650" n.d.)

before the bitcoin created in them is available to be spent, unlike non-coinbase transactions, which require only six confirmations, or six blocks added to the block chain after the block in which they are included. Originally, this reward was set to 50 BTC; currently, the reward is 25 BTC. A transaction fee is included in the block information, but in most transactions that amount is far less than the amount created in the coinbase transaction. ("Bitcoin Developer Guide" n.d.). This coinbase transaction amount is set to reduce by half every 210,000 blocks validated, approximately every four years. Bitcoin core also has a hard set amount of total bitcoin that will ever be in existence of 21,000,000 BTC, estimated to be reached in the year 2140 (Nielson 2013).

To create a unique identifier for this transaction block, we concatenate six elements of the information above: the *version number*, the *hash of the previous block*, the *Merkle root*, the *timestamp*, the *difficulty*, and the *nonce* (Table. 2.2, 2.3) (Shirriff 2014). Together, these six elements are known as the *block header*. When the hash of a transaction block is found, it is actually the hash of the block header. This hash is the unique identifier for this block (Antonopoulos 2014).

The *version number*, 2, is the version of Bitcoin core utility used when this block was created. This lets all users in the Bitcoin network know the rules in place when this block was created and hashed (Nielson 2013). Using older versions of Bitcoin core can

lead to forks in the block chain (“Bitcoin Developer Guide” n.d.). The *hash of the previous block* is the result of hashing the header from the previous transaction block.

The *Merkle root* is a special hash of all of the transactions included in this transaction block and is derived from the ‘leaves’ of the Merkle tree (Fig. 2.4) , with H_A referring to the hash of transaction A. Transactions make up the leaves of this Merkle tree. To find the root, the transactions to be included in the new block are hashed, separated into pairs, and each pair is hashed using a double-SHA256 function. The results from each pair’s hash are then paired and hashed, and so on, until only a single 256 bit hash remains, the Merkle root.

The purpose of hashing the transactions into a Merkle root is to provide a unique identifier. Once all of the transactions are double-hashed, any attempt to change any information included in those transactions will cause the root to completely change, providing a level of security within the block. A transaction may be confirmed to be included in a transaction block by showing that it is included in the Merkle tree for that

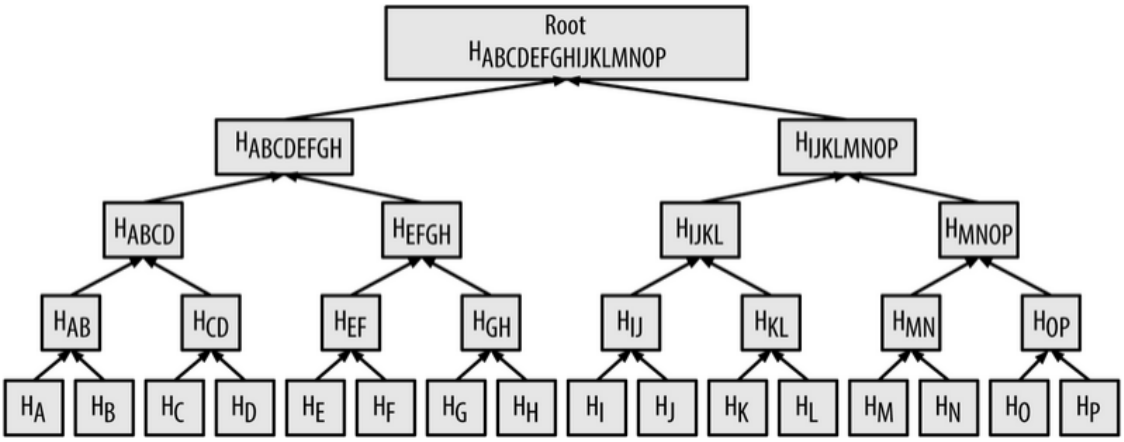

Figure 2.4 (Antonopoulos 2014)

block. To do this, a node would acquire records of the other transactions, or the hashes of the other transactions, included in this block, hash the transaction under investigation in with them, and compare the resulting hash with the Merkle root in the block header (Antonopoulos 2014). For example, to verify transaction J is included in this block, acquire or recreate the hashes ABCDEFGH, MNOP, KL, and I, from other nodes, complete the rest of the branches of the Merkle tree, and compare this current hash with the Merkle root. If they match, the proposed transaction J is confirmed to be included in this transaction block ("Bitcoin Developer Guide" n.d.).

The *timestamp* indicates when the block is created. The purpose of the timestamp is to provide an official time when the transactions in this block first existed. Any transactions in a later block that include the same bitcoin as inputs as this block includes as inputs are rejected. This is determined by comparing the transaction identifier to those in the transaction pool and those included in the block chain. Remember, each bitcoin input has a unique identifier, which changes when it becomes an output. From the current block, only the output may be included as an input in a later block (Antonopoulos 2014). The timestamp is the key that prevents spending of the same bitcoin multiple times. The timestamp proves when an unspent transaction output is included as an input in a new transaction and prevents that same bitcoin or that same transaction from being included in a block that is timestamped later (Nakamoto 2008).

The *difficulty*, or *target difficulty*, is a measure of how difficult the current target makes it to successfully mine a block. A difficulty of 1 would mean that it would be very easy to successfully mine a block, requiring a minimal number of calculations. The difficulty at the time block #350650 was mined was 46,717,549,644.71, or 46 billion

times more difficult than mining a block would be at a difficulty of 1. Difficulty is also used in calculating the *target*, or maximum value of a successfully mined block's hash (Antonopoulos 2014). More details on target and difficulty, and how they relate to proof-of-work, are discussed later.

The *nonce* is a random number included in the block header. Miners will try different random numbers in the header until they find one which causes the hash of the header to meet the proof-of-work requirement set out by Bitcoin core. If it takes too long to find a nonce, miners may change either the timestamp, extending it up to two hours after the actual time, or the coinbase transaction and begin searching for a nonce once again (Antonopoulos 2014).

Before a block is validated, nodes must provide a *proof-of-work*. That is, the node is required to provide proof that it has solved a computationally difficult puzzle before the rest of the network will accept the validation of the transaction. This proof-of-work makes it so that dishonest nodes which try to falsify previous blocks must work harder than honest nodes that are trying to extend the block chain, and it throttles the creation of new bitcoin ("Bitcoin Developer's Guide" n.d.). All nodes and miners will work on the same block's proof-of-work at the same time. This constitutes a race, as each node works on a solution to the proof-of-work in order to claim the prize: the transaction fee and the newly created bitcoin. Since nodes are made up of groups of miners, the winnings are divided among miners in a manner determined by each node. The puzzle is to find a random number (the nonce) such that, when it is included in the transaction header, the resulting hash includes a predetermined number of leading zeroes in hexadecimal, thus falling below the target number. The successful miner will have attempted around 16^{10}

or approximately 10^{12} different nonce values before discovering one that gives the desired result (Antonopoulos 2014).

The proof-of-work *target* is determined automatically by each node based on specifications in Bitcoin core. At the time of this writing, the current target and the calculated *difficulty* in reaching it is given in Table 2.5. The target may also be said to be 16 leading zeroes in hexadecimal (Table 2.5). The target is determined by dividing the maximum hash value by the difficulty. Per Bitcoin core, difficulty is recalculated every 2016 blocks, or approximately every two weeks, based on the average amount of time

Decimal Target
5452275669824046697205997511035633087075590495334196838 40
Hexadecimal Target
00000000000000000163C710000000000000000000000000000000000000000000
Difficulty
49446390688.241

Table 2.5 Current Target and Difficulty
("DecimalTarget" n.d., "HexTarget" n.d., "GetDifficulty" n.d.)

between block validations over the previous 2015 blocks. The last 2015 blocks are used instead of the last 2016 due to an error in Bitcoin core ("Bitcoin Developer Guide" n.d.). The intended validation time is approximately 10 minutes between blocks. If the previous average was less than 10 minutes, the difficulty would increase, meaning the number of leading zeroes would increase so the target would decrease. If the previous average was greater than 10 minutes per transaction, the difficulty would decrease, so fewer leading zeroes and a higher target. The new difficulty is calculated by dividing the actual total time it took to calculate the last 2015 blocks by 20160 minutes, then multiplying this

$$\text{New Difficulty} = \frac{(\text{Total Time for Last 2015 blocks})}{(10 * 2016)} * \text{Old Difficulty}$$

result by the old difficulty. The target difficulty is based on CPU power, not the number of bitcoin spent (Antonopoulos 2014).

Once the nonce is found, the block is broadcast across the Bitcoin network and becomes part of the *block chain*. Nodes will only accept the new block if all transactions are valid. As each node receives and validates the block, they stop working on this block and begin work on the next new block, thus indicating their acceptance of the block. The hash of this block becomes the previous hash in the next new block's header (Fig. 2.5) (Antonopoulos 2014).

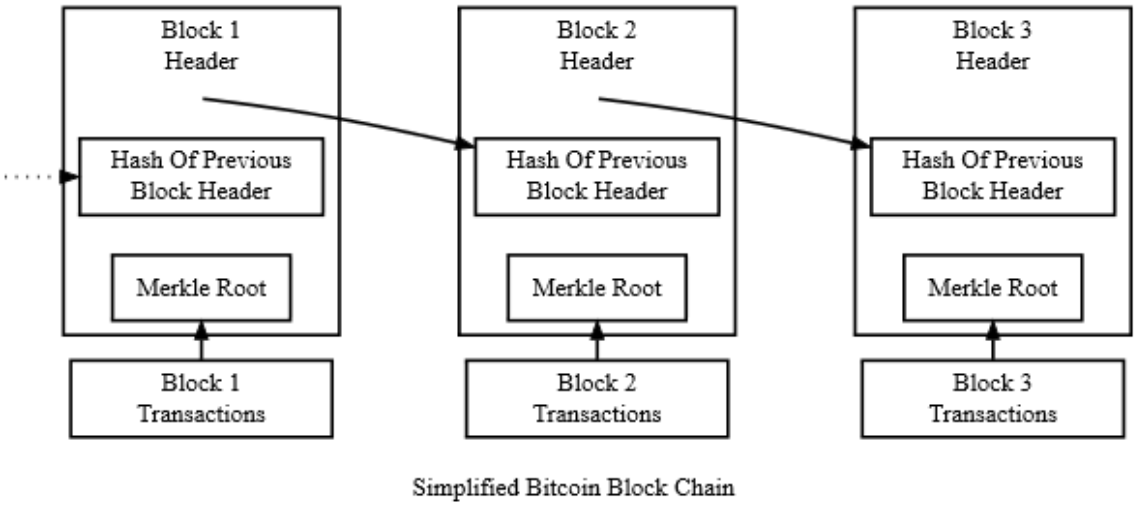

Figure 2.5 ("Bitcoin Developer Guide" n.d.)

The block chain is Bitcoin's public ledger. Each new block contains the hash of the previous block's header, linking all transaction blocks in turn back to the genesis block, or block #0. By chaining the blocks together in this manner, changing a transaction becomes impossible to accomplish without altering every transaction after it in the chain ("Bitcoin Developer's Guide n.d.). Each node stores a record of all blocks validated by that node. Multiple nodes that contain the same record are said to be in consensus (Antonopoulos 2014).

The *block number* indicates the block height, or number of blocks since the genesis block. When a fork occurs, two blocks or more with the same height will exist until the main block chain is once again revealed (Fig. 2.6). Forks most often occur when

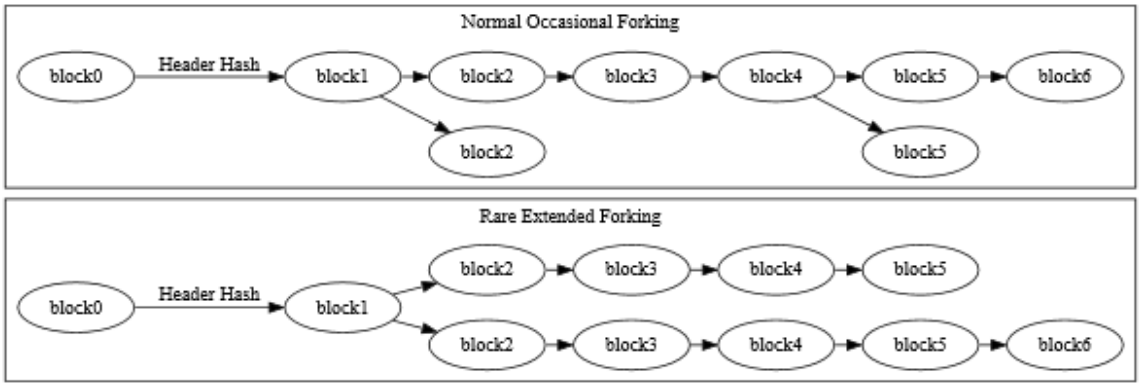

Figure 2.6 (Bitcoin Developer Guide n.d.)

two nodes broadcast their proof-of-work solution at the same time. Other nodes will receive the broadcast first from the node closest to them, accept it as valid, and begin working on the next block based on whichever block they received. Soon, another block will be validated, confirming the block that its node had previously received, extending the block chain on one side of the fork. Once this proof-of-work is broadcast, all nodes drop their work on the second block regardless of which fork it stemmed from and begin work on a third block (Antonopoulos 2014).

Another way forks occur is when a new version of Bitcoin core is released and some nodes update before others. New blocks are created, following either the new rules and violating the old or following the old rules and not implementing new, causing a fork. Eventually, the fork controlled by the updated nodes should extend further than the other, causing that side of the fork to be accepted by more nodes, thus becoming the main block chain. Regardless of what caused the fork, the stale block from the shorter side of the fork becomes an orphan block and is no longer part of the block chain (Antonopoulos

2014). Any non-coinbase transactions that are part of this orphaned block are not lost. Nodes that were trying to add to the orphaned block will know which transactions have been included in the main chain's new block once they receive and validate it. The coinbase transaction is lost, as the block it was the claimed prize for is now defunct. Recent updates in Bitcoin core make it improbable that any fork will ever be longer than a single block, so the coinbase transaction of an orphaned block will never have enough confirmations to be spent ("Bitcoin Develop Guide" n.d.).

3 DIGITAL KEYS

Public key cryptosystems use a pair of keys, one public and one private, to secure information. Public keys, or encrypting keys, may be made available to the public. Without the private key, or decrypting key, discovery of the original message is a very difficult endeavor, far more difficult than the original encryption was using the public key. To implement a public key cryptosystem, user *A* chooses a public key E by following an agreed upon set of rules and makes this public key available for all to use while keeping both his private key and how the public key was constructed a secret. Anyone sending *A* a message m will encrypt the message using *A*'s public key, with a result $s = E(m)$. *A* receives the encrypted message s and applies his private key D to it to open up the original message, $D(s) = D\big(E(m)\big) = m$ (Rosen 2011).

Public keys are also used in creating digital signatures. Digital signatures are used to verify the authenticity of a message; that is, digital signatures are used to prove a message actually came from the correct person and that none of the data in the message has been altered since the message was created. The private key and a version of the original message is used to create the signature. Since $D\big(E(m)\big) = m = E\big(D(m)\big)$, the public key may be used to decrypt the signature, verifying the origin of the message (Rosen 2011).

Bitcoin uses digital keys to prove ownership of bitcoin and digital signatures to reliably transfer bitcoin from one wallet to another within the Bitcoin network. When a user sets up a new wallet, the wallet software creates a private key. While this private key is kept secret, any public keys are calculated based on this key. In Bitcoin, the public key hash is the user's wallet address (Antonopoulos 2014). When a user spends his or her

bitcoin, s/he applies his/her digital signature to the previously unspent bitcoin. This signature both confirms that no transaction data has been altered since the bitcoin was received and verifies that the user is authorized to spend this bitcoin (Antonopoulos 2014). The creators of Bitcoin needed to choose which public key cryptographic system to implement.

Public key cryptography was introduced in the late 1970s in two main formats: the Diffie-Hellman key exchange algorithm in 1976 and the more popular RSA algorithm in 1977 (Rosen 2011).

Simple RSA Example:

1. Choose two distinct prime numbers $p = 7$ and $q = 19$
2. Compute $n = pq$ $n = 7 \times 19 = 133$
3. Compute the totient of the product $\varphi(n) = (p-1)(q-1) = 108$
4. Choose any number $1 < e < 108$ that is coprime to 108. $e = 17$
5. Compute the modular multiplicative inverse d of e

$$17d \equiv 1 \ (mod\ 108) \qquad d = 89$$

public key $(n, e) = (133,17)$ private key $d = 89$

encryption function $c(m) = m^e \bmod n = m^{17} \bmod 133$

decryption function $c^{-1}\big(c(m)\big) = \big(c(m)\big)^d \bmod n = \big(c(m)\big)^{89} \bmod 133$

where *m* is the message to be encrypted.

Simple Diffie-Hellman Example:

1. Choose a prime number $p = 103$
2. Find a primitive root of p $r = 5$
3. Choose two private keys $k_1 = 27$ and $k_2 = 31$

4. Compute the common key $K \equiv r^{k_1 k_2} \pmod{p} \equiv 90,\ 0 < K < p$

5. Compute the public keys $94 = y_1 \equiv r^{k_1} \pmod{p},\ 0 < y_1 < p$

$40 = y_2 \equiv r^{k_2} \pmod{p},\ 0 < y_2 < p$

The decryption function for RSA, $c^{-1}\big(c(m)\big) = \big(c(m)\big)^d \bmod n = \big(c(m)\big)^{89} \bmod 133$ works because of Euler's theorem. That is,

From above, $c^{-1}\big(c(m)\big) = \big(c(m)\big)^d = (m^e)^d = m^{ed}$

Since $ed \equiv 1\ (mod\ \varphi(n))$, $ed = k\varphi(n) + 1$ for some integer *k*.

So $m^{ed} = m^{k\varphi(n)+1} = m\big(m^{\varphi(n)}\big)^k$.

By Euler, if $\gcd(m, n) = 1$, $m^{\varphi(n)} \equiv 1(mod\ n)$,

then $c^{-1}\big(c(m)\big) = m^{ed} \equiv m\ (mod\ n)$.

And, by Chinese Remainder Theorem, since $pq = n$, if $p|m$,

Then $c^{-1}\big(c(m)\big) = m^{ed} \equiv m\ (mod\ p)$.

(Works similarly for q.)

Both of these algorithms rely upon the difficulty of the mathematics involved to undo them (Rosen 2011).

As computers become more powerful, the ability to break encryption using large primes becomes easier, requiring the use of larger primes. As the primes become larger, the limited computing power of common electronics, such as cell phones and tablets, requires an algorithm for digital signatures which uses very little computing power to encrypt and still requires a massive amount of computing power to decrypt when the private key is unknown. In 1985, the use of elliptic curves was proposed as an alternative to the use of large primes, factoring, and logarithms (Sullivan 2013).

Elliptic curve cryptography allows for low computing power for implementing security but requires massive amounts of computing power in a brute force attempt to break (Sullivan 2013). According to Rosen, "An elliptic curve is the set of points (x, y) that satisfy $y^2 = x^3 + ax + b$ where a and b are real numbers."(2011) For the following example, x and y are real numbers. New points on the curve can be constructed using known points. Given two points on the curve, P and Q, a third point, R, may be found on the curve by addition, $P + Q = R$. Two primary methods can be used to accomplish this (Rosen 2011).

For the first method, *point addition*, let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$, with $P \neq \pm Q$. The point R can be seen by first drawing a line l through P and Q which will intersect the curve at a third point, R'. R is the reflection of R' across the *x*-axis and the result of adding the points P and Q, so $R' = (x_3, -y_3)$ and $R = (x_3, y_3)$ (Fig. 3.1). This result may also be found algebraically (Rosen 2011).

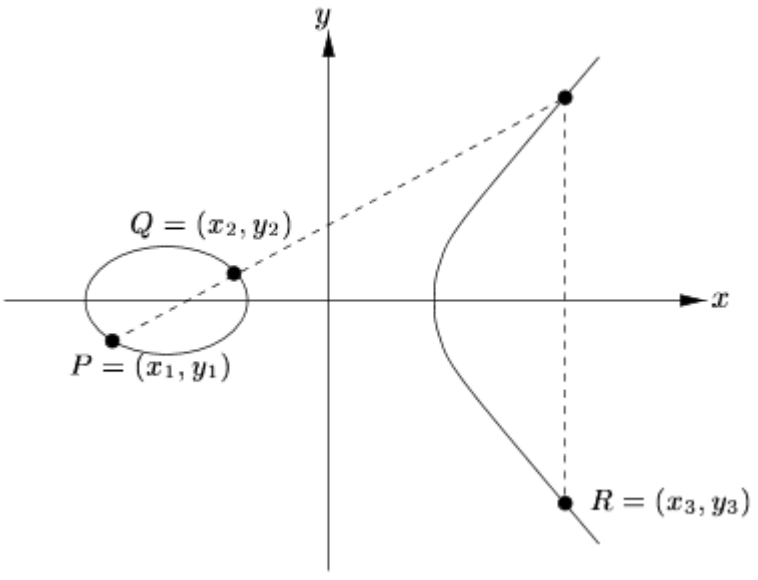

Figure 3.1 Point Addition (Johnson, et al 2001)

To find R, first find the slope of line l given by $m = \frac{y_2 - y_1}{x_2 - x_1}$, and the equation of the line given by $y = m(x - x_1) + y_1$. Hence, by substitution:

$$(m(x - x_1) + y_1)^2 = x^3 + ax + b$$

$$y_1^2 + 2my_1(x - x_1) = x^3 - m^2(x - x_1)^2 + ax + b$$

Since the sum of the roots of a cubic function equals the negative of the coefficient of the squared term, we know $x_1 + x_2 + x_3 = m^2$, then $x_3 = m^2 - x_1 - x_2$. To find the value of y_3, we solve $-y_3 = m(x_3 - x_1) + y_1$ so $R' = (x_3, -y_3)$, which is reflected across the x-axis to find R (Johnson, Menezes, Vanstone 2001).

For the second method, *point doubling*, $P = Q$, $P \neq -Q$. To find the line l, note that as Q gets closer to P, the slope of the line through P and Q gets closer to the slope of the tangent line to P. Once $P = Q$, $2P = R$, and $R = (x_3, y_3)$ is the reflection of R' across the x-axis (Fig. 3.2). The slope of line l, found by implicit differentiation of the curve at P, is given by $m = \frac{(3x_1^2 + a)}{2y_1}$. The coordinates of R are found in a similar manner

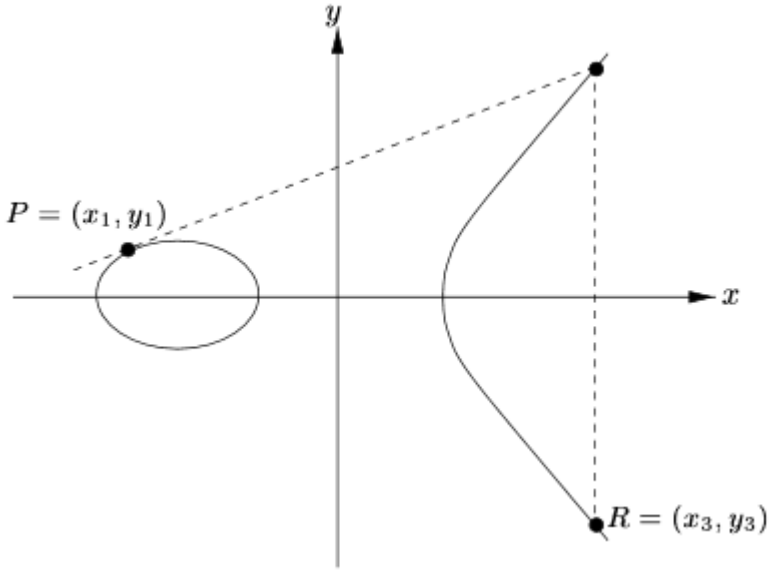

Figure 3.2 Point Doubling (Johnson 2001)

to those in point addition: $x_3 = m^2 - 2x_1$ and $y_3 = m(x_1 - x_3) - y_1$ (Rosen 2011).

As an example, consider the elliptic curve $y^2 = x^3 + x + 4$ over the finite field F_{23}, with $P = (7, 3)$ and $Q = (15, 17)$ points on the curve (Johnson 2001). First, an example of point addition, followed by point doubling:

Point Addition: $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{17-3}{15-7} = \frac{7}{4}$ $P = (x_1, y_1), Q = (x_2, y_2)$

To find m mod 23:

$4m \equiv 7 \bmod 23$ (has a unique solution since $\gcd(4, 23) = 1$)

$6 \cdot 4 \equiv 6 \cdot 7 \bmod 23$ (since $24 \equiv 1 \bmod 23$)

$24m \equiv 42 \equiv 19 \bmod 23$

So $m = 19$.

To find R: $x_3 = m^2 - x_1 - x_2$

$= (19)^2 - 7 - 15 = 339 \equiv 17 \bmod 23$

$y_3 = m(x_1 - x_3) - y_1$

$19(7 - 17) - 3 = -193 \equiv 14 \bmod 23$

So $R = (x_3, y_3) = (17, 14)$

Point Doubling: $m = \left(\frac{3x_1^2 + a}{2y_1}\right) = \frac{3(7)^2+1}{2(3)} = \frac{74}{3}$ $P = (x_1, y_1)$

To find m mod 23:

$3m \equiv 74 \equiv 5 \bmod 23$

$8 \cdot 3m \equiv 8 \cdot 5 \bmod 23$

$24m \equiv 40 \equiv 17 \bmod 23$

So $m = 17$.

To find R: $x_3 = m^2 - 2x_1$

$$= (17)^2 - 2(7) = 275 \equiv 22 \bmod 23$$

$$y_3 = m(x_1 - x_3) - y_1$$

$$= 17(7 - 22) - 3 = -258 \equiv 18 \bmod 23$$

So $R = (x_3, y_3) = (22, 18)$

Multiplication on the elliptic curve works like repeated point doubling (Antonopoulos 2014).

Point addition and doubling of the elliptic curve defined in the secp256k1 standard as $y^2 = x^3 + 7$ over the prime field F_p where $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$ is the method used by Bitcoin to create Bitcoin wallet addresses, or public key hashes (Antonopoulos 2014) (Fig. 3.3). To generate a Bitcoin address, or a wallet address, a user chooses a random number, typically generated by a good random number generator, as their private key, k. To generate a public key, first multiply the private key, k, by a predetermined generator point, G, defined by the secp256k1 standard, which lies on the curve $y^2 \bmod p = (x^3 + 7) \bmod p$. The result is another point on the curve, or the public key, K (Antonopoulos 2014) (Fig 3.4).

This public key is still not in the form of a wallet address. To find the wallet address, the coordinates of the public key K are concatenated, their SHA256 hash is computed, then the RACE Integrity Primitives Evaluation Message Digest (RIPEMD) hash is computed from the SHA256 hash, or $A = RIPEMD160(SHA256(K))$. This produces a 160-bit number, A (Antonopoulos 2014).

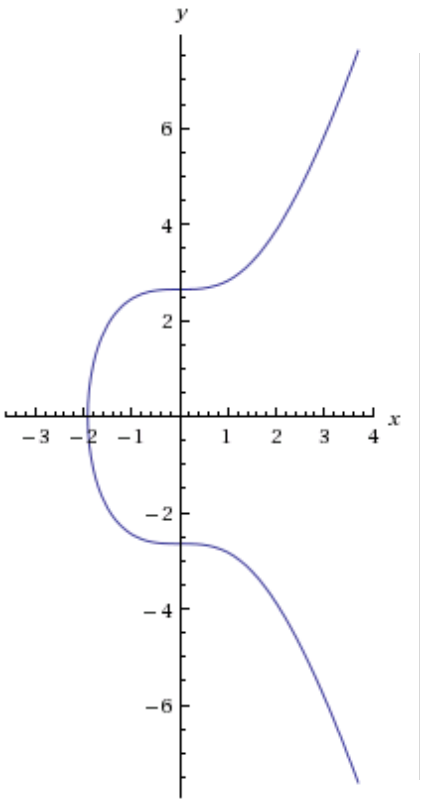

Figure 3.3 Elliptic Curve: $y^2 = x^3 + 7$

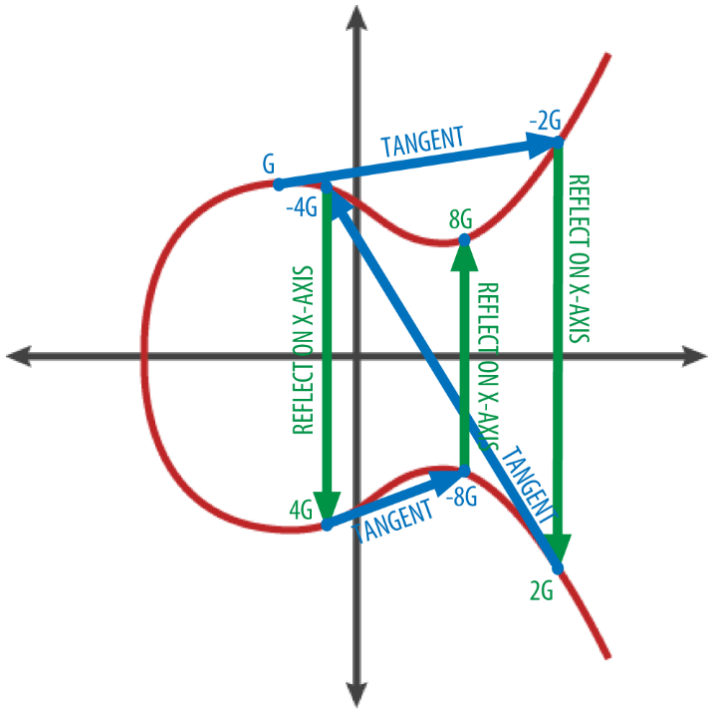

Figure 3.4 $K = kG$ (Antonopoulos 2014)

Alice requires Bob's public key hash, or wallet address, to send him the bitcoin she owes him for the e-book (Fig. 3.5). Once she receives it, Alice includes Bob's public key hash as part of an encumbrance she places on the transaction. This encumbrance is a set of instructions that will only allow this output, this bitcoin from Alice, to be spent if the spender, Bob, proves that he controls the private key that produces Bob's public key hash. Bob's wallet now includes this payment from Alice as a spendable output ("Bitcoin Developer Guide" n.d.).

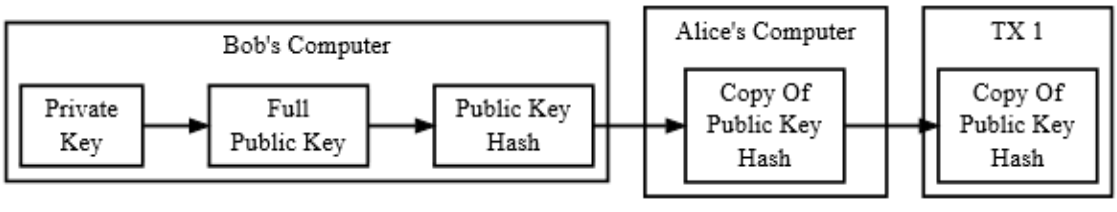

Figure 3.5 Creating a Public Key Hash to Receive a Payment ("Bitcoin" n.d.)

Bob decides to spend this bitcoin when making a purchase from Carl. Bob includes the bitcoin from Alice as an input in a transaction. To satisfy the encumbrance placed by Alice, Bob must prove that he controls the private key that produces the public key hash that Alice included in the encumbrance. So, before he broadcasts the transaction, Bob signs it. By signing the transaction, Bob locks the transaction contents, creating a layer of security in the transaction ("Bitcoin Developer Guide" n.d.). When he signs, he includes his full unhashed public key, *K*, and his digital signature. His digital signature is created using his private key, *k*, by encrypting a hash of the transaction contents. Bitcoin uses Bob's public key to decrypt his digital signature, verifying that the hash of the transaction contents remains unaltered and that Bob controls the private key associated with his public key (Johnson, et al 2001). If either of the keys used aren't Bob's, or the hashes of the transaction contents don't match, the signature is rejected and the bitcoin remains in Bob's wallet (Fig. 3.6) ("Bitcoin Developer Guide" n.d.).

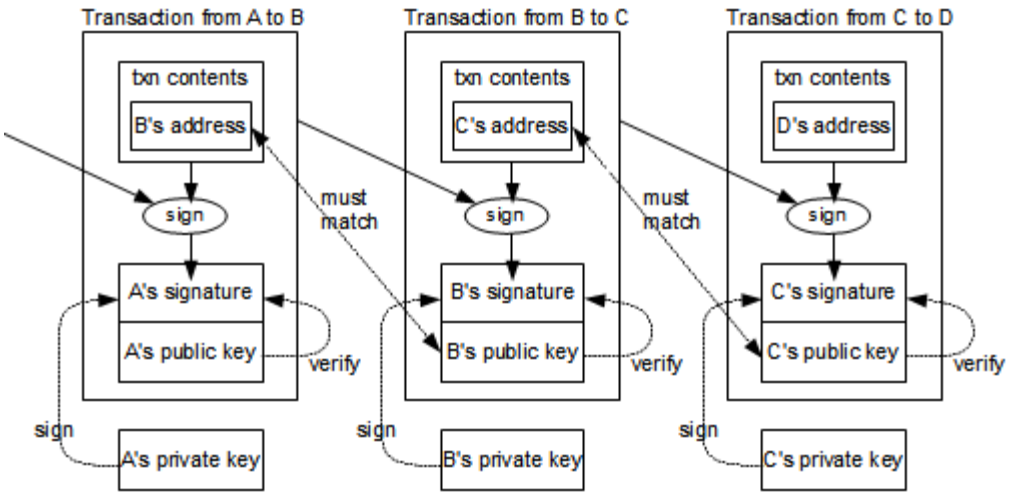

Figure 3.6 Bitcoin Transaction Signature Chain (Shirriff "Bitcoins" 2014)

4 SECURE HASH ALGORITHM 256

Secure hash algorithms are “iterative, one-way hash functions that can process a message to produce a condensed representation called a *message digest”* (“FIPS” 2012, p. 3), or hash. Once processed through the hash function, any change in a message or download results in a different digest or hash. Hence, these algorithms may be used to validate the integrity of a message or download. Digital signatures, message authentication codes, and random numbers may be generated and authenticated in this manner (“FIPS” 2012).

Each algorithm following the secure hash standard includes a preprocessing stage and a hashing stage. During the preprocessing stage, the data is padded to an appropriate size, divided into blocks, and each block is divided into words. SHA256 can handle messages of size 2^{64} or less. Its block size is 512 bits, so the data to be hashed is padded to a bit size divisible by 512. Each block consists of 16 32-bit words (“FIPS” 2012). The hashing stage is described in detail below. The resulting hash has eight 32-bit words, for a total of 256 bits. To illustrate the process, I’ve run the word ‘cryptography3’ through the SHA256 hash function by hand, verified by an Excel worksheet created by David Rabahy (2014).

Preprocessing of the data takes place in which the original data is converted, using the ASCII table, into binary, then to hexadecimal and concatenated. The bit ‘1’ is then appended to the original message. Then append the number of ‘0’ bits required to bring the message length, modulo 512, to 448 bits. Finally, append the length of the original message, without the padding, as 64 bit integer, making the entire length of the message, in bits, a multiple of 512 (“FIPS” 2001).

For a Bitcoin transaction, this gives 1024 bits of information to be hashed. Each SHA256 run will take 16 32-bit chunks, or *words*, denoted w_i where $i = 0, \ldots, 63$, for a total of 512 bits each run (Table 4.1).

w_0	cryp	6	3	7	2	7	9	7	0
		0110	0011	0111	0010	0111	1001	0111	0000
w_1	togr	7	4	6	f	6	7	7	2
		0111	0100	0110	1111	0110	0111	0111	0010
w_2	aphy	6	1	7	0	6	8	7	9
		0110	0001	0111	0000	0110	1000	0111	1001
w_3	3	3	3	8	0	0	0	0	0
		0011	0011	1000	0000	0000	0000	0000	0000
w_4		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_5		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_6		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_7		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_8		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_9		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_{10}		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_{11}		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_{12}		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_{13}		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_{14}		0	0	0	0	0	0	0	0
		0000	0000	0000	0000	0000	0000	0000	0000
w_{15}		0	0	0	0	0	0	6	8
		0000	0000	0000	0000	0000	0000	0110	1000

Table 4.1 Words 0 - 15 in Hex and Binary

To continue preparing the data, the first 16 words are used to determine the remaining 48 words, for a total of 64 words. For words w_i, $i \geq 16$,

$$w_i = (w_{i-16}) + (w_{i-7}) + \left[(w_{i-15})_{(7)} + (w_{i-15})_{(18)} + (w_{i-15})_{(3)}\right] + [(w_{i-2})_{(17)} + (w_{i-2})_{(19)} + (w_{i-2})_{(10)}]$$

where each subscript (j) is the number of binary digits the given word is shifted to the right, the first two shifts wrap back to the beginning, the third is truncated ("FIPS" 2001). Table 4.2 illustrates how to find word w_{16} (Table 4.2).

	$\left[(w_1)_{(7)} + (w_1)_{(18)} + (w_1)_{(3)}\right]$																													
w_1	0	1	1	1	0	1	0	0	0	1	1	0	1	1	1	1	0	1	1	0	0	1	1	1	0	1	1	1	0	0	1	0
	e				4				e				8				d				e				c				e			
7	1	1	1	0	0	1	0	0	1	1	1	0	1	0	0	0	1	1	0	1	1	1	1	0	1	1	0	0	1	1	1	0
	d				9				d				c				9				d				1				b			
18	1	1	0	1	1	0	0	1	1	1	0	1	1	1	0	0	1	0	0	1	1	1	0	1	0	0	0	1	1	0	1	1
	0				e				8				d				e				c				e				e			
3	0	0	0	0	1	1	1	0	1	0	0	0	1	1	0	1	1	1	1	0	1	1	0	0	1	1	1	0	1	1	1	0
	3				3				b				9				a				f				3				b			

	$[(w_{14})_{(17)} + (w_{14})_{(19)} + (w_{14})_{(10)}]$								
w_{14}	0	0	0	0	0	0	0	0	
	0000	0000	0000	0000	0000	0000	0000	0000	

w_0	6	3	7	2	7	9	7	0
w_9	0	0	0	0	0	0	0	0
$\left[(w_1)_{(7)} + (w_1)_{(18)} + (w_1)_{(3)}\right]$	3	3	b	9	a	f	3	b
$[(w_{14})_{(17)} + (w_{14})_{(19)} + (w_{14})_{(10)}]$	0	0	0	0	0	0	0	0
w_{16}	9	7	2	c	2	8	a	b

Table 4.2 Finding Word 16

Words 17 through 63 are found similarly.

Now the hash function begins.

For the first round, begin with eight constants, h_0 through h_7, determined by the NSA, the creators of SHA256. These constants are derived from the square roots of the first eight prime numbers as follows (Table 4.3).

Primes (p)	$\sqrt{p}$	$mod\ 1$	$int((\sqrt{p}\ mod\ 1) * 16^8)$	Hexadecimal
2	1.414213562	0.414213562	1779033703	6a09e667
3	1.732050808	0.732050808	3144134277	bb67ae85
5	2.236067977	0.236067977	1013904242	3c6ef372
7	2.645751311	0.645751311	2773480762	a54ff53a
11	3.316624790	0.316624790	1359893119	510e527f
13	3.605551275	0.605551275	2600822924	9b05688c
17	4.123105626	0.123105626	528734635	1f83d9ab
19	4.358898944	0.358898944	1541459225	5be0cd19

Table 4.3 Beginning Constants

Each of these constants is A through H of our initial run through. Next, each of the initial values is translated into binary (Table 4.4).

A	6 0110	A 1010	0 0000	9 1001	E 1110	6 0110	6 0110	7 0111
B	B 1011	B 1011	6 0110	7 0111	A 1010	E 1110	8 1000	5 0101
C	3 0011	C 1100	6 0110	E 1110	F 1111	3 0011	7 0111	2 0010
D	A 1010	5 0101	4 0100	F 1111	F 1111	5 0101	3 0011	A 1010
E	5 0101	1 0001	0 0000	E 1110	5 0101	2 0010	7 0111	F 1111
F	9 1001	B 1011	0 0000	5 0101	6 0110	8 1000	8 1000	C 1100
G	1 0001	F 1111	8 1000	3 0011	D 1101	9 1001	A 1010	B 1011
H	5 0101	B 1011	E 1110	0 0000	C 1100	D 1101	1 0001	9 1001

Table 4.4 Constants to Binary

By looking at the bits of A, B, and C, find the *Majority value*, Ma. To do this, for each bit position, if more of the bits are 0 the output is 0, and if more are 1 the output is 1. Then, translate the majority value into hexadecimal (Table 4.5).

A	6				a				0				9				e				6				6				7			
	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
B	b				b				6				7				a				e				8				5			
	1	0	1	1	1	0	1	1	0	1	1	0	0	1	1	1	1	0	1	0	1	1	1	0	1	0	0	0	0	1	0	1
C	3				c				6				e				f				3				7				2			
	0	0	1	1	1	1	0	0	0	1	1	0	1	1	1	0	1	1	1	1	0	0	1	1	0	1	1	1	0	0	1	0
Ma	0	0	1	1	1	0	1	0	0	1	1	0	1	1	1	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
	3				a				6				f				e				6				6				7			

Table 4.5 Majority Value

To find Σ0 from A, first shift the bits of A two bits to the right. Then shift the bits of A 13 bits to the right. Then shift the bits of A 22 bits to the right. Now sum the bits in each position. If the sum is even, enter a 0, and if the sum is odd, enter a 1. Then translate Σ0 from binary to hexadecimal. The first bit from A is highlighted in each row (Table 4.6).

A	6				a				0				9				e				6				6				7			
	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1
>																																
2	1	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0	1	1	1	1
2	0	0	1	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1	0	1	0	0	0
Σ0	c				e				2				0				b				4				7				e			
	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	1	0	1	1	0	1	0	0	0	1	1	1	1	1	1	0

Table 4.6 *Σ0*

Next, compute the *Choosing value*, Ch, on E, F, and G, by looking at the bits in E. If the bit in E is 0, choose the bit of the same position in G. If the bit in E is 1, choose the bit of the same position in F. Convert the choosing to hex. The appropriate bits based on the position in E are highlighted (Table 4.7).

E	5				1				0				e				5				2				7				f			
	0	1	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1
F	9				b				0				5				6				8				8				c			
	1	0	0	1	1	0	1	1	0	0	0	0	0	1	0	1	0	1	1	0	1	0	0	0	1	0	0	0	1	1	0	0
G	1				f				8				3				d				9				a				b			
	0	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	1	1	0	0	1	1	0	1	0	1	0	1	1
Ch	0	0	0	1	1	1	1	1	1	0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1	0	0	0	1	1	0	0
	1				f				8				5				c				9				8				c			

Table 4.7 Choosing Value

To find Σ1 from E, first shift the bits of E six bits to the right. Then shift the bits of E 11 bits to the right. Then shift the bits of E 25 bits to the right. Now sum the bits in each position. If the sum is even, enter a 0, and if the sum is odd, enter a 1. Then translate Σ1 from binary to hexadecimal. The first bit from E is highlighted in each row (Table 4.8).

E	5				1				0				e				5				2				7				f			
	0	1	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1
>6	1	1	1	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	1
11	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0	1	0	1	0
25	1	0	0	0	0	1	1	1	0	0	1	0	1	0	0	1	0	0	1	1	1	1	1	1	1	0	1	0	1	0	0	0
Σ1	0	0	1	1	0	1	0	1	1	0	0	0	0	1	1	1	0	0	1	0	0	1	1	1	0	0	1	0	1	0	1	1

Table 4.8 $\Sigma 1$

Now compute the *First Sum* by adding the hexadecimal numbers for Σ0, the Choosing, H, our first 32 bit chunk of input denoted W_t, and the NSA defined constant for each round K_t, where $t = 0$ thru 63 is the round we are working on (Table 4.9). These constants K_t are derived using the equation $\text{int}((\sqrt[3]{p_t} \bmod 1) * 16^8)$ where p_t is one of the first 64 prime numbers.

W_0	6	3	7	2	7	9	7	0
K_0	4	2	8	a	2	f	9	8
H	5	b	e	0	c	d	1	9
Ch	1	f	8	5	c	9	8	c
Σ1	3	5	8	7	2	7	2	b
First Sum	5	6	e	a	6	6	d	8

Table 4.9 First Sum

Define new values A through H. For our new A, add Σ0, the Majority, and the First Sum (Table 4.10).

	Σ0	c	e	2	0	b	4	7	e
	Ma	3	a	6	f	e	6	6	7
First	Sum	5	6	e	a	6	6	d	8
New	A	5	f	7	b	0	1	b	d

Table 4.10 New A

For our new E, add the old D and the First Sum (Table 4.11).

Old	D	a	5	4	f	f	5	3	a
First	Sum	5	6	e	a	6	6	d	8
New	E	f	c	3	a	5	c	1	2

Table 4.11 New E

For the rest, each letter moves down one in line. So Old A becomes New B, Old B becomes New C, and so on until Old G becomes New H, and Old H just drops off.

This process continues through a total of 64 rounds. The final step to produce the hash is to add the original constants, h_0 through h_7, to the final values for A though H, producing 8 32-bit words in hexadecimal (Table 4.12).

h'_0	c	2	6	6	a	5	3	8
h'_1	6	f	d	9	e	b	9	3
h'_2	3	0	c	f	e	0	b	c
h'_3	e	a	3	2	c	3	1	6
h'_4	1	1	5	7	8	e	b	1
h'_5	d	b	9	f	a	5	7	2
h'_6	4	6	0	0	6	c	1	f
h'_7	b	6	6	2	6	3	9	6

Table 4.12 Final Words

These words are concatenated to get the result of our hash in hexadecimal (Table 4.13).

c266a5386fd9eb9330cfe0bcea32c31611578eb1db9fa57246006c1fb6626396

Table 4.13 ‘cryptography3’ Hash

However, since Bitcoin transaction blocks include more than 512 bits, but less than 1024, the process begins again for the remaining 16 blocks of information, using the final h'_0 through h'_7 in place of the constants originally used (Table 4.11). After all of the information has been included, we end with 8 32-bit blocks, completing one SHA256 hash. After 128 rounds, we’ve successfully compressed 1024 bits of information to 256 bits. Remember, for Bitcoin, each transaction block is put through a double hash for a total of 192 rounds.

Above, the word ‘cryptography3’ was hashed once by the SHA256 algorithm. Now, ‘cryptography2’ is hashed once to compare (Table 4.14). This entry changes the

24dabb19cd16be8dce985ec10847cc2e7e38634fb50c263593d865d134e9077

Table 4.14 'cryptography2' Hash

input by a single binary digit, but the resulting hash is entirely different. Thus, if any of the transaction data is altered by even a single bit, the entire hash result is altered. This demonstrates the high level of security built into Bitcoin to ensure the trust network remains intact.

5 SUMMARY

Bitcoin employs the SHA256 hash function and elliptic curve cryptography to ensure security and reinforce trust. This trust is critical to bitcoin survival because of the decentralized nature of a digital currency. Traditional currencies rely upon banking systems and governments for mediation, requiring only trust in the financial institutions and government regulation. Bitcoin, on the other hand, relies upon the consensus of its users. Use of cryptographic security prevents tampering in every level of a transaction, from inception through fulfillment. To fully understand the basis of this trust requires an investigation of the mathematics used to secure bitcoin transactions and the block chain. Additional topics for exploration of the mathematics of bitcoin would include alternative signing protocols to the more commonly used one described here and statistical modeling of the probability of successfully altering transaction data.

GLOSSARY

bit - binary digit, either ‘0’ or ‘1’

bitcoin (BTC) – the name of the cryptocurrency

bitcoin – the network and the software

block – a group of transactions, marked with a timestamp and a hash of the previous block

block chain – a list of validated blocks, each linked to the previous block, all the way to the genesis block

block header – the six elements of a block that are hashed to compute the proof-of-work; they include the version of bitcoin being used at the time the block is created and for all transactions in the block, the hash of the previous block, the Merkle root, the timestamp, the number of bits in the block, and the nonce.

coinbase transaction – constructed by a miner or node, this is the reward for successfully mining the block

concatenate – attach a data string to another data string end to beginning with no spaces

confirmations – including a transaction in a block is one transaction; once another block is mined on the same block chain, the transaction has two confirmations, etc.

difficulty – a network wide setting that controls how much computation is required to produce a proof-of-work

genesis block – the first block in the block chain, used to initialize the cryptocurrency

hash – values returned by a hash function, such as SHA256

miner – a Bitcoin user who uses special software to find valid proof-of-work for new blocks and is issued a certain number of bitcoins in exchange

node – a cooperative collection of bitcoin miners which form a full client that owns the block chain and is sharing blocks and transactions across the network

proof-of-work – solution to a computationally difficult puzzle that is very difficult to solve but easy to confirm; in Bitcoin, miners find a numeric solution to the SHA256 algorithm that meets the difficulty target

satoshi – the smallest allowed subdivision of a bitcoin, or 0.00000001 BTC

target – difficulty target; the difficulty at which all the computation in the network will find blocks approximately every 10 minutes

transaction – a public entry in the block chain which records a signed data structure expressing a transfer of value

wallet – an electronic record of all private and public keys which are used to sign transactions, proving ownership of specific bitcoin

REFERENCES

Antonopoulos, Andreas M. *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*. O'Reilly Media, 2014. Accessed February 20, 2015. http://chimera.labs.oreilly.com/books/1234000001802/index.html

"Bitcoin Developer Guide." *Bitcoin.* n.d. Accessed October 6, 2014. https://bitcoin.org/en/developer-guide.

“Block #350650.” *Blockchain Info.* n.d. Accessed April 3, 2015. https://blockchain.info/block/000000000000000001450e1fcca7d74877361c1e154ee0e888c211a212b4f4382.

Bryans, Danton. "Bitcoin and Money Laundering: Mining for an Effective Solution." *Indiana Law Journal* 89, no. 1 (2014): 441-72. Accessed October 14, 2014. http://web.ebscohost.com.emporiastate.idm.oclc.org/ehost/detail/detail?sid=6e53e4ae-22e9-4cd1-a83a-9ec5dc1e4654@sessionmgr4005&vid=0&hid=4104&bdata=JnNpdGU9ZWhvc3QtbGl2ZQ==#db=lgh&AN=93745150.

"DecimalTarget." *Bitcoin Block Explorer*. n.d. Accessed April 3, 2015. http://blockexplorer.com/q/decimaltarget.

"FIPS 180-4: Secure Hash Standard (SHS)." *NIST: Computer Security Division.* March, 2012. Accessed March 6, 2015. http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf.

"GetDifficulty." *Bitcoin Block Explorer*. n.d. Accessed April 3, 2015. http://blockexplorer.com/q/getdifficulty.

Greenberg, Andy. "Crypto Currency." *Forbes.* April 20, 2011. Accessed February 6, 2015. http://www.forbes.com/forbes/2011/0509/technology-psilocybin-bitcoins-gavin-andresen-crypto-currency.html.

"HexTarget." *Bitcoin Block Explorer*. n.d. Accessed April 3, 2015. http://blockexplorer.com/q/hextarget.

Johnson, Don, Alfed Menezes, and Scott Vanstone. "The Elliptic Curve Digital Signature Algorithm (ECDSA)." Certicom. January 1, 2001. Accessed February 16, 2015. http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf.

Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." *Satoshi Nakamoto Institute.* October 31, 2008. Accessed October 20, 2014. http://nakamotoinstitute.org/bitcoin/.

Nielsen, Michael. "How the Bitcoin Protocol Actually Works." *DDI: Data-driven Intelligence.* December 6, 2013. Accessed February 6, 2015. http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/.

Rosen, Kenneth H. "Cryptology." In *Elementary Number Theory and Its Applications*. 6th ed. Reading, Massachusetts: Addison-Wesley Pub., 2011.

Rabahy, David. *SHA-256*. October 9, 2014. Accessed March 6, 2015. https://docs.google.com/a/g.emporia.edu/spreadsheets/d/1mOTrqckdetCoRxY5QkVcyQ7Z0gcYIH-Dc0tu7t9f7tw/edit#gid=1194752368

Shirriff, Ken. "Mining Bitcoin with Pencil and Paper: 0.67 Hashes per Day." *Ken Shirriff's Blog.* September 2014. Accessed March 6, 2015. http://www.righto.com/2014/09/mining-bitcoin-with-pencil-and-paper.html?m=1.

Shirriff, Ken. "Bitcoins the Hard Way: Using the Raw Bitcoin Protocol." *Ken Shirriff's Blog.* February 2014. Accessed March 6, 2015. http://www.righto.com/2014/02/bitcoins-hard-way-using-raw-bitcoin.html

Sullivan, Nick. "A (Relatively Easy to Understand) Primer on Elliptic Curve Cryptography." *ARS Technica.* October 24, 2013. Accessed February 6, 2015. http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/.

Velde, Francois. "Bitcoin: A Primer." *Chicago Fed Letter*, no. 317 (2013). Accessed October 17, 2014. https://www.chicagofed.org/publications/chicago-fed-letter/2013/december-317.

Wei, Dai. "Bmoney." *Wei Dai.* 1998. Accessed February 5, 2015. http://www.weidai.com/bmoney.txt.

I, Sophia Crossen, hereby submit this thesis/report to Emporia State University as partial fulfillment of the requirements for an advanced degree. I agree that the Library of the University may make it available to use in accordance with its regulations governing materials of this type. I further agree that quoting, photocopying, digitizing or other reproduction of this document is allowed for private study, scholarship (including teaching) and research purposes of a nonprofit nature. No copying which involves potential financial gain will be allowed without written permission of the author. I also agree to permit the Graduate School at Emporia State University to digitize and place this thesis in the ESU institutional repository.

_____
Signature of Author

_____
Date

_____
Title of Thesis

_____
Signature of Graduate School Staff

_____
Date Received