# Bitcoin Anonymity and The Block Chain

Cian Burns

Masters in Computer Science (MCS)

University of Dublin, Trinity College

Supervisor: Dr Donal O'Mahony

Submitted May 2014

# Declaration

I, Cian Burns, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signature:

Date:

## Summary

Bitcoin is a peer to peer digital cryptocurrency that has risen in popularity in recent times. The Bitcoin protocol relies on a structure known as the Block Chain. This Block Chain is the public ledger of transactions that have occurred over the entire history of Bitcoin and is constantly growing. I will address many aspects of Bitcoin from the history of Bitcoin mining to some in-depth discussions on many of the controversies that plague Bitcoin.

The aim of this dissertation is to demonstrate that by using certain properties of a Bitcoin transaction that it is possible to extract additional information about a user and their addresses from the Block chain. I will discuss the implementation of a system capable of extracting this information and I will also outline the design of a system that would be capable of utilising this implementation to provide a service to both the Bitcoin community and a financial regulator.

As a proof of concept I will demonstrate how using this system it is possible to analyse the actions of a given user. I will use the Bitcoin exchange Mt. Gox as a case study and I will attempt to analyse what really happened to their users missing Bitcoin, by utilising the tools I will create.

# Acknowledgements

There are many people who without their support this dissertation would not have been made possible. First I would like to thank my supervisor Dr Donal O'Mahony for his constant support and advice over the course of the last few months. I would like to thank my friends and classmates for all of their assistance over the last five years of study. Lastly I would like to thank my parents, my brother and my girlfriend for their continued support over the years. Without these people none of this would have been possible.

# Table of Contents

# Table of Figures

# 1. Introduction

Bitcoin is a digital peer to peer based crypto currency that was created in 2009 by Satoshi Nakamoto. [1] Since its creation it has grown in popularity and is now accepted as a payment method on many online stores. At the heart of the Bitcoin network is the Block chain, the public ledger of transactions. In this dissertation I aim to show how, using certain properties of Bitcoin transactions, it is possible to extract additional information about a user and their transaction history from the Block chain.

I will discuss the operation of Bitcoin including the Blockchain, the authorisation of Bitcoin transactions through mining and some of the venerability's of Bitcoin. I will then compare Bitcoin with some other crypto currencies. I will talk about the main differences between the currencies which mainly revolve around how they authorise transactions and mint new coins. I will also focus on some of the most important media stories that have affected Bitcoin since its creation in 2009. These will range from Bitcoin thefts to companies trying to reduce the influence of Bitcoin within their platform. I will also discuss some more controversial happenings such as the closure of the largest Bitcoin exchange "Mt. Gox" and the FBI raid on the Bitcoin black market website; "The Silk Road". In this same chapter I will also discuss some of the technologies relevant to the system design I will outline later.

I will close this section by discussing one of the motivations for this project, Taxing and Regulating Bitcoin. As I will mention later there are no tools to assist either financial regulators or members of the Bitcoin community with this task. The system that I outline in the design section would be capable of aiding both parties with extracting the information they need from the Block chain. I will discuss the design of this system and mention some of the technologies I would use to fully implement the system. I will then talk about the details of the portion of this system that I implemented as a proof of concept. The portion that I aim to implement consists of the back end scripts responsible for the data processing part of the application. I will not be implementing the front end and some other components that I will discuss in my design chapter due to time and resource constraints.

I will bring the dissertation to a conclusion by demonstrating how this system can be used to audit a user. I will demonstrate this by performing a partial analysis of the transactions connected to the Bitcoin exchange, Mt. Gox. From this analysis I will offer my opinion on what happened in the controversy over the missing Bitcoin that Mt. Gox had been holding. I will

finally discuss the conclusions of the project and present my opinion on the future of the Bitcoin network. I will also outline some of the potential future work to continue on from my implementation.

The aim of this dissertation is to help both users and regulators to better understand the data publicly available in the Block chain by designing a system that is capable of monitoring the Bitcoin Block chain in near real time and allows users to query an address to view its transaction history. The system could be used by tax regulators to monitor compliance or to help when a user needs to declare their Bitcoin in their tax returns by generating a comprehensive transaction history. The system also could be used to get more information about a user before a transaction is carried out and facilitate ratings of transactions once the transaction has taken place.

# 2. State of the Art

Bitcoin is a digital currency that was first described in a 2009 paper by Satoshi Nakamoto a pseudonym for a yet unidentified computer scientist. [1] Satoshi begins by pointing out the issue of the current digital commerce system. The current digital commerce system relies on a trusted third party such as a credit card company or payments company like PayPal to mediate the transaction. Satoshi states that the problem with this is that it requires trust and an unnecessary third party. He offers Bitcoin and its cryptographic proof of work as an alternative solution to the problem of conducting e-commerce. In his paper he describes Bitcoin as a digital currency authorized by a decentralized distributed peer to peer network.

He goes on to then describe the individual parts of the protocol and network beginning with coins and transactions. He defines a coin as a chain of signatures that prove its ownership and a transaction is the process of signing this coin so that it belongs to the recipient. Transactions take place between two users by sending Bitcoins from one address to another. Transactions are made by the sender signing the coins over to the receiver creating a chain of signatures representing the coins. The details of these transactions are broadcast publicly to the entire Bitcoin network. Transactions are then signed by the network in a process called mining. Mining involves each node in the Bitcoin network attempting to compute the hash of a group of transactions. Once this has been computed the transactions are authorized by the network and are added to the public transaction log known as the Block chain. The Block chain can be queried to check if a user has any coins and to see who they have sent coins to before. During this process the peer to peer network fulfils the roll of a mint by issuing new currency and the roll of a bank authorising transactions between users.

## 2.1. Bitcoin Block Chain

The Bitcoin network is a peer to peer network that is made up of two types of nodes, Full nodes and Miners. Full nodes are responsible for keeping the peer-to-peer network running. When a transaction is created or a new block mined it is the full nodes responsibility to broadcast this data to any machine that has connected to it. The miners are responsible for acting as a mint in the Bitcoin network. They authorise transactions and issue new coins in a process known as mining. There are 6 steps that miners of the network must carry out for a transaction to be fully accepted into the Block chain.

> *"1) New transactions are broadcast to all nodes.*
>
> *2) Each node collects new transactions into a block.*
>
> *3) Each node works on finding a difficult proof-of-work for its block.*
>
> *4) When a node finds a proof-of-work, it broadcasts the block to all nodes.*
>
> *5) Nodes accept the block only if all transactions in it are valid and not already spent.*
>
> *6) Nodes express their acceptance of the block by working on creating the next block in the chain, using the hash of the accepted block as the previous hash." [1]*

The Bitcoin network relies on a central Block chain that is available for anyone to inspect. This Block chain is effectively a public ledger for the Bitcoin network. The peer to peer network uses this Block chain in its decisions while processing transactions. When Alice wants to send Bitcoin to Bob she must publish their transaction to the network for it to be authorised and added to the Block chain. There are several steps involved in this exchange. First Alice must acquire Bob's public address, this will be denoted as $A_{Bob}$. Once Alice knows $A_{Bob}$ it can now begin the process of telling the network that it is transferring some coins into that address. It starts by Alice deriving $PubK_{Bob,}$ the public key associated with Bobs address. Alice then hashes the previous transaction that sent the coins to her with $PubK_{Bob}$ and then signing the output. This creates a chain of signatures that represents an amount of Bitcoin. These signatures are the networks way of proving ownership of a coin. [2] It is possible for Bob to verify that the coins came from Alice by checking the signature using Alice's public key, $PubK_{Alice}$.

This signed hash is then broadcast to the Bitcoin network. This is how Alice tells the network that she is sending the coins to Bob. The transaction will be sent to as many nodes in the network as possible in the hope that one of them will add the transaction to a block and sign it to say that it is valid. Once this happens, the next time Bob starts his wallet app it will catch up with what happened while he was offline and he will then see that Alice sent him some Bitcoin. When a node receives a confirmed block of transactions it must first check that they are all legal transactions. An illegal transaction can simply be a user who doesn't have any coins pretending they have some and trying to send them to another address. A transaction like this would be spotted when a transaction is included in a block and broadcast to the Bitcoin network. A miner would see the new block and then check each transaction against the Block chain by looking at each transactions input reference. This is a reference to the previous

transaction in the chain, the transaction that gave the input coins. If all the transactions are valid then the block is accepted by the miners. By checking the Block chain they can find out how many Bitcoins are owned by the individual address. If the address does not show up before the attempted spend in the Block chain, then it is concluded that the address has no Bitcoin to spend and the transaction will be rejected and not included in the mining of the next block.

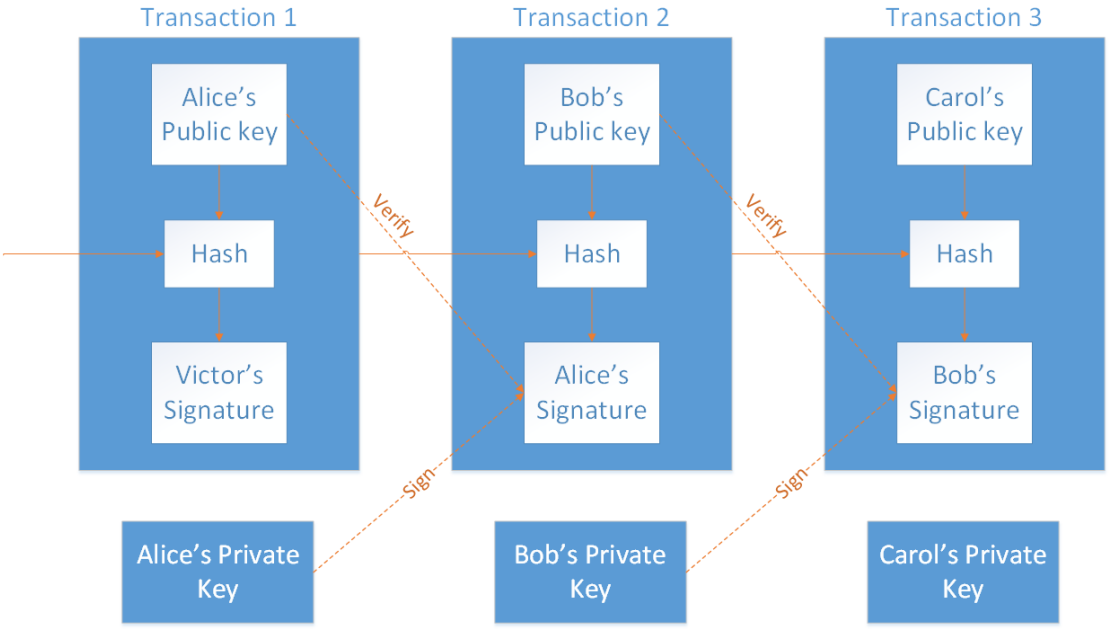


*Figure 1 Example chain of Transactions Between three users.*

A more malicious illegal transaction is known as a Double Spend attack. Should Alice wish to double spend her coins, she would need to send her coins to Bob and Carol at the same time by forwarding her transaction to two different nodes in the peer to peer network. Since the Bitcoin network is a distributed peer to peer based network it is possible to send a different message to two distant nodes and due to propagation delay there would be a period where they will both think they have the correct message. By doing this Bob would be able to see that the Alice has announced to the network that she is sending coins to Bob, likewise Carol would see that there have been coins sent from Alice to Carol. One important thing to note is that these transactions have yet to be signed into a block. The attack occurs where both Bob and Carol think the coins have been sent to them. However to prevent something like this happening the Bitcoin network uses a proof of work concept for singing off on blocks of potentially valid transactions. This proof of work will prevent two nodes on opposite side of the Bitcoin network from signing two transactions like the ones above effectively double spending Alice's coins. Until a transaction has been signed into a block and accepted by the network it is dangerous to assume that it is guaranteed to complete, in the case above only one transaction would be valid.

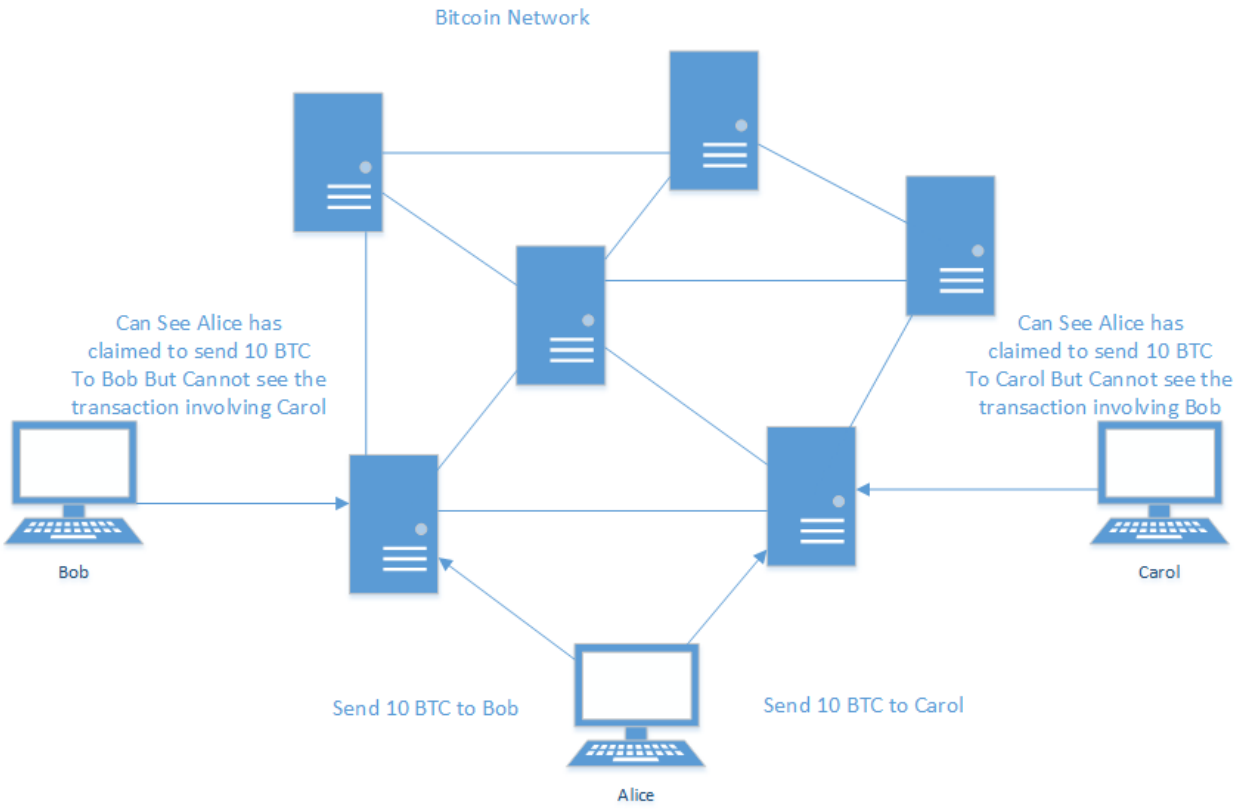


*Figure 2 Example of a double spend attack*

## 2.2. Hash functions in the Block chain

Hash functions were created in an effort to manipulate an input data set into a unique output set. [3] Functions capable of this were useful for many uses such as checksums and hash tables. Hash functions are used as checksums by hashing an input file and sending the file and the hash to the recipient. Since hash functions will always give the same output for the same input the user can then hash the file when it is received. By comparing the two hashes the user can then verify if the file has been tampered with in transit.

Hash functions can also be used to scramble and obscure data. They can take in a string of any length and perform various mathematical operations upon the data causing it to become scrambled. There are some key properties that all hash functions should have. All hash functions are designed to have these properties however over time due to increasing computation power and mathematical study they are sometimes shown to be venerable. First they should non-reversible. This means given the output, H, of a hash function it should be very difficult to calculate M, the message that when hashed produces H. If a hash function has this property then it is known to be pre-image attack resistant.

```
Hash(M)=H        Func(H)!=M
```

The second property a hash function should have is that given a message M1 it should be very difficult to compute M2 such that `Hash(M1)=Hash(M2)`. This is known as second pre-image attack resistant and differs slightly from the third property of a hash function. The third property of a hash function is that it should be collision resistant. This means that it should be difficult to find two messages M1 and M2 that hash to the same value. The difference between collision resistance and second pre-image attack resistance is that in a second pre-image attack one message of M1 or M2 is known while in a collision, neither M1 nor M2 is known. [4]

To produce a collision the node hashes as many of the unconfirmed transactions as possible to create the Merkle root, and then hashes the Merkle root with the hash of the last block from the Block chain and a nonce. The Merkle root is calculated by hashing transactions in a tree pattern. The transactions are the leaf nodes and each branch node is the hash of its two child nodes. The aim is to obtain one root hash containing the hash of all the transactions in that block. [5] The nonce is a randomly generated number that is chosen to cause the output hash to begin with a specified number of zero bits. The number of zeroes is determined by the difficulty level of the current block. The difficulty is set by the peer to peer network to keep the frequency of new blocks being mined at roughly once every 10 minutes. Once found, the resulting hash is known as a block and is appended to the network's Block chain. The Block chain is the public record of all Bitcoin transactions that have taken place since the very first coins were created by Satoshi.

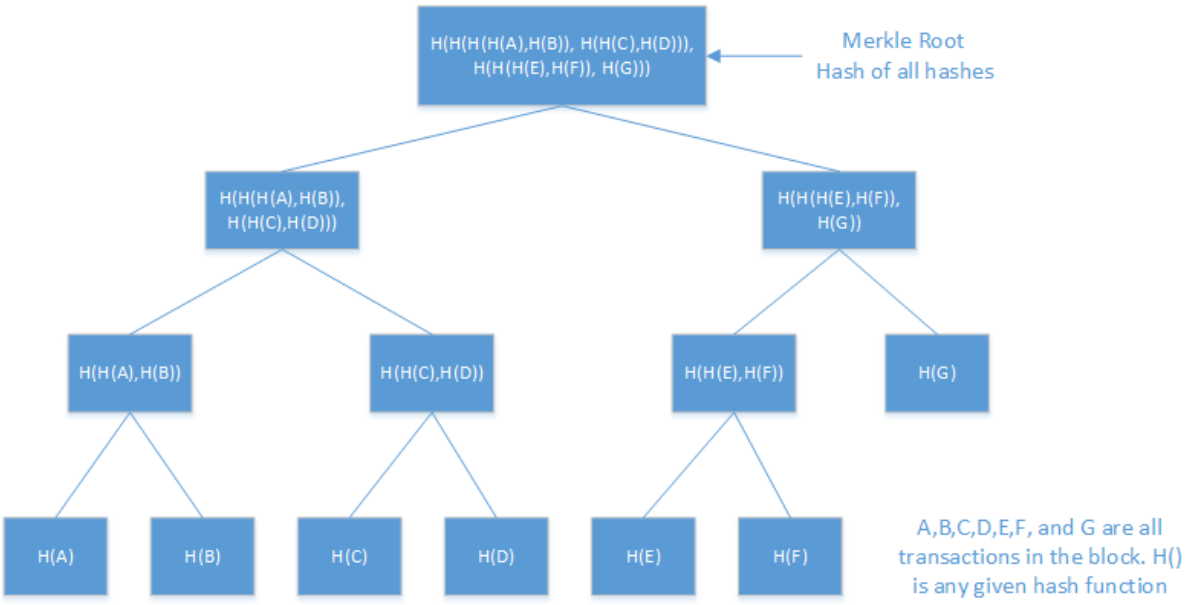


*Figure 3 Merkle Tree*

Satoshi also discusses how using a Merkel tree allows for disk space to be saved when storing coins. By using a Merkel tree it is only necessary to store the most recent transaction and the interior hashes in the tree. The rest of the leaf nodes and hashes can be "Pruned" in an effort to save disk space. The paper describes how it is possible to verify if a transaction is valid or not based on this Merkle root. By recalculating the Merkle root of the block the transaction claims to be a member of it is possible to verify if the transaction is genuine or not.

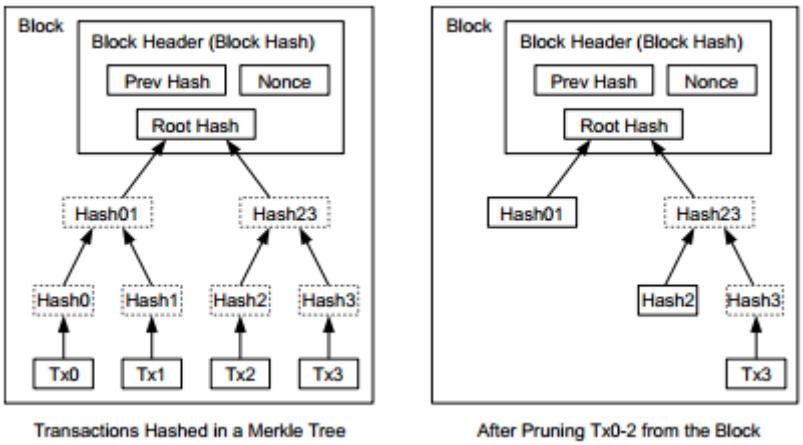


*Figure 4 Merkle tree pruning* [1]

Satoshi points out that there would need to be a way of making sure that the coins have not already been spent and this is where he introduces the concept of using a proof of work scheme. He describes this proof of work process as follows:

> *"The proof-of-work involves scanning for a value that when hashed, such as with SHA-256, the hash begins with a number of zero bits"*

To compute the proof of work a node must calculate a hash collision using SHA-256, the second generation of the Secure Hash Algorithm. The SHA-256 algorithm belongs to a group of cryptographic hash functions known as the SHA-2 family of functions. It was first defined by the National Security Agency of America in 2001. The SHA-2 family are named after the output size of the corresponding functions, in the case of SHA-256 it has an output digest of 256 bits. [6] The SHA-2 family of algorithms were created to overcome some of the short comings of previous encryption algorithms that had been used. Message Digest 5 or MD5 for example was created in 1992 as the successor to the MD4 function and became widely used as a hash algorithm. However by 1996 work on breaking the MD5 algorithm had progressed to the point where many cryptographers were now recommending the use of a stronger algorithm like SHA-1. [7] In 2005 SHA-1 was beginning to be considered broken as some analysis showed that it may not be secure enough for future use. Due to this analysis the American National

Institute of Standards and Technology required many government applications to move to use SHA-2. [8]

The process of signing blocks is known as mining. Mining involves signing a block and having the output signature begin with a set number of zero bits. Since it is not possible to predict the output of the SHA-256 hash algorithm the only way to get a collision is by attempting a brute force calculation changing a nonce for each calculation. This is a very compute heavy task as it is a non-deterministic process due to the unpredictable nature of the SHA-256 algorithm [9]. A node must test every random number in the hope of generating a collision. This random number is known as the nonce. In the case of the Bitcoin protocol a hash collision is when the output of the mining process begins with a given number of zero bits. The number of zero bits is defined by the difficulty of the current block, a value that is calculated by the network every thousand blocks. There is an incentive in the form of a reward of Bitcoins for the node that solves the block. This originally started at 50 Bitcoins and is currently at 25 Bitcoins. Satoshi built into the protocol that the reward will halve after a certain number of blocks. It will halve every 210,000 blocks which is roughly every two years. On top of the reward for finding the solution of the block the node the finds the correct solution also gets a cut of the transaction fee. At the time of writing this the default transaction fee in the Bitcoin-QT software created by the Bitcoin Foundation is 0.0001 BTC or roughly $0.04. [10]

To mine a block a node must compute the correct hash of a block header. This block header consists of six components. The Bitcoin protocol version that is being used, the timestamp of when the hash was started, the difficulty of the current block, the hash of the previous block, the chosen nonce, and the Merkel root. The Merkel root is the hash of all transactions that have been authorised by the node attempting to mine the block. This means that the transactions are not hashed in the solution to the block and are instead hashed indirectly through the Merkel root. [11] The Merkel root consists of the hash of the transactions that are being accepted in this round. This keeps the size of the header small while allowing nodes in the network to quickly verify what transactions were in that block. The Merkel root consists of a tree of hashes. The leaf nodes in the tree are the individual transactions and each of the parent nodes in the tree are the hashes of the two nodes below it. This leads to one root node that is the hash of all hashes of the transactions being accepted in this block.

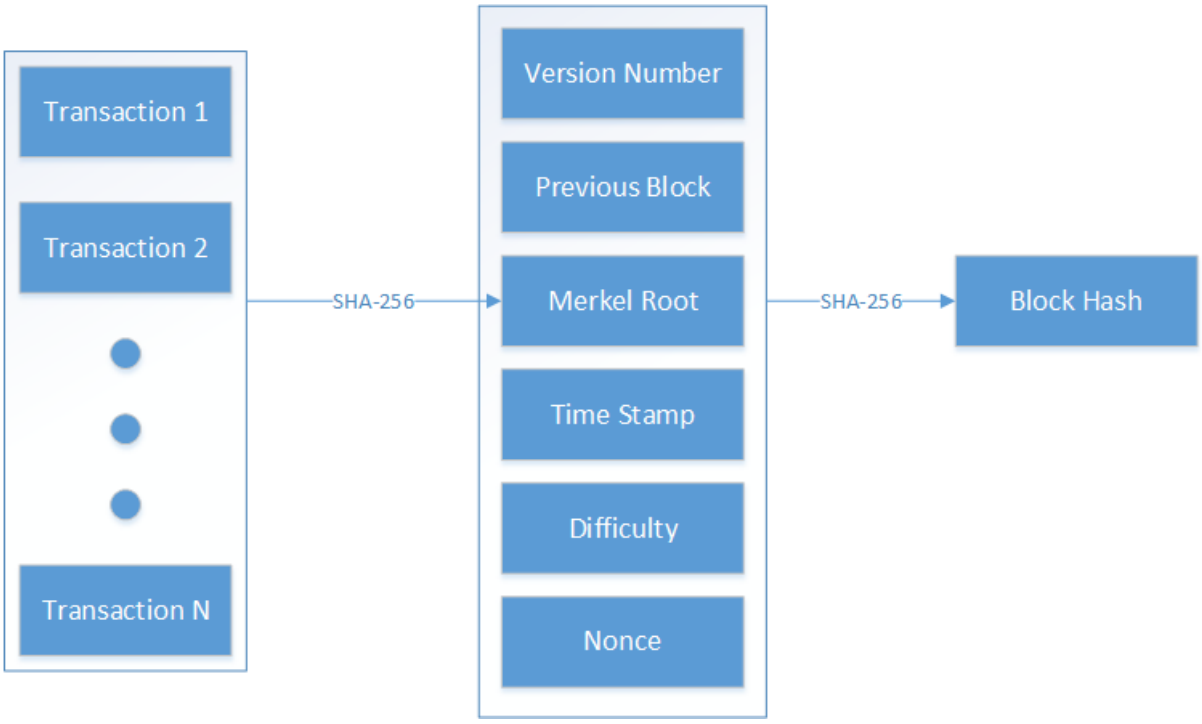


*Figure 5 The Transaction Hashing Process*

## 2.3. Mining ASICs

In 2013 M.B. Taylor from the University of California wrote a paper about the inception of Bitcoin ASIC miners and how they became the only profitable means of Bitcoin mining. [12] He outlines the evolution of Bitcoin mining hardware from general purpose CPU mining, through GPU and FPGA mining right up to modern day ASIC mining. As with all papers on Bitcoin the author describes how the Bitcoin network functions. However the author's approach differs in that he separates it out into the apparent functionality to a given group of users as opposed to discussing the functionality from a more generalised perspective. He first talks about how Bitcoin works for an average user looking to use Bitcoin to trade or purchase goods and services. He discusses public and private keys, Bitcoin wallets, transaction fees, and the Bitcoin Block chain. In the second part of this section he addresses how the network functions from a Miner's perspective. He discusses the two types of reward that a miner gets for solving the block, the block reward and the mining fee. The miner's fees are optional fees that a user can add to their transaction in an effort to get it noticed faster. Miner's fees are collected by the miner that solves the block. This tends to mean that the higher the miners fee on a transaction the higher the chance of it being included in a block by a miner.

The author discusses the history of Bitcoin mining. He begins with Satoshi mining on his CPU back in 2009. He notes that this is no longer profitable and mining on a CPU would actually cost you money. He also adds a graph that shows the price per Gigahash of power the network has. It can be seen from this graph the need for technologies to evolve as the price per Gigahash dropped over the years. This led to the eventual rise of GPUs and their highly parallel architectures in July of 2011 when CPU mining was no longer profitable. In the graph the horizontal lines represent the price per Gigahash of the corresponding technology. Some common mining applications for Graphics cards are OpenCL Miner and CUDAMiner. The later of these two is specifically for NVidia graphics cards. Both applications use the highly parallel architecture of graphics cards to compute hundreds of hashes in an effort to mine a block. In the paper he lists the NVidia GTX570, a mid to high range graphics card released in 2010, as being able to achieve 155 Megahashes per second. There are many resources available for calculating the estimated hash output of graphics cards online. [13]

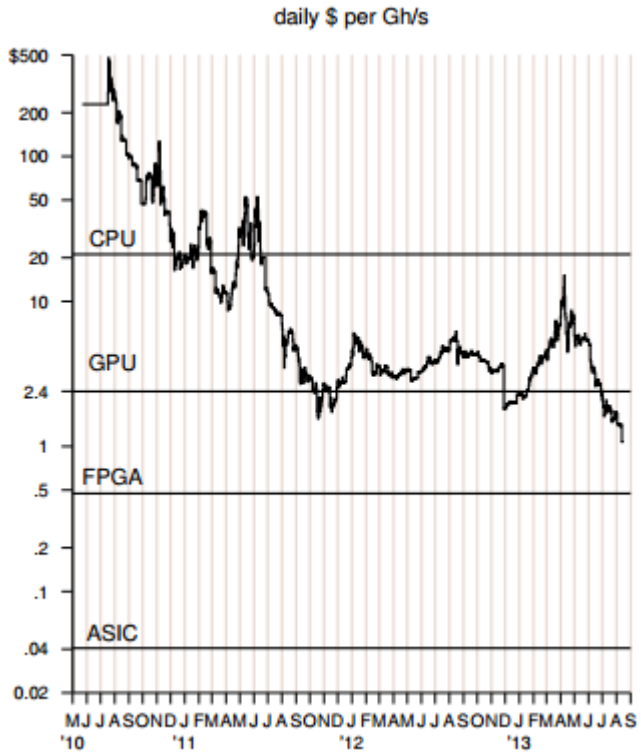


*Figure 6 Price per Gigahash* [12]

M.B. Taylor also analyses the cost efficiency and the associated drawbacks with a GPU based mining rig. These vary from space and cooling issues, to the lack of PCI-e ports on many consumer grade motherboards necessary to connect a GPU to a system. Controlling many of these factors is possible but they in turn increase the cost per Gigahash of the final system. The author then move on to discuss how well Field Programmable Gate Arrays or FPGAs can

handle the various parts of the SHA-256 algorithm. He notes that FPGAs excel at many of the bitwise operations necessary for SHA-256 when compared to even high end GPUs. Even with this disadvantage the author lists several FPGA boards and their performances, which perform better than some of the Graphics cards noted earlier in the paper. Due to the fact that GPUs were still competitively priced and had similar mining capabilities over the FPGAs, it can be seen in the graph above that when FPGAs running the Bitcoin algorithm appeared in 2011 GPUs managed to still remain profitable. This profitability lasted until mid-2013 when Butterfly labs announced their first Application Specific Integrated Circuit.

When Butterfly labs released their first generation of ASICs, they were capable of producing several Gigahashes per second as opposed to Megahashes for GPUs and FPGAs, and came at prices rivalling many high end graphics cards. One of their systems called the Jalapeño was priced at $150 and was capable of producing over 4 Gigahash a second. Not only are these ASICs more powerful than GPUs and CPUs they are also more power efficient thanks to their fixed logic and lower clock rates. [14] Soon after two more companies, ASIC Miner and Avalon, released their own ASICs. With such a large selection of ASICs available in the market GPU and FPGA mining was no longer feasible. The author then notes that this technology will need to continue to scale to provide even better Gigahash rates. He also notes that this may not be possible due to an issue affecting general purpose CPUs, a problem known as Dark Silicon or what some people call the Power Wall. As the transistor size decreases, the energy needed to power the chips increases as more transistors are squeezed onto the chips die. This scaling can lead to issues with heat dissipation and power consumption. [15]

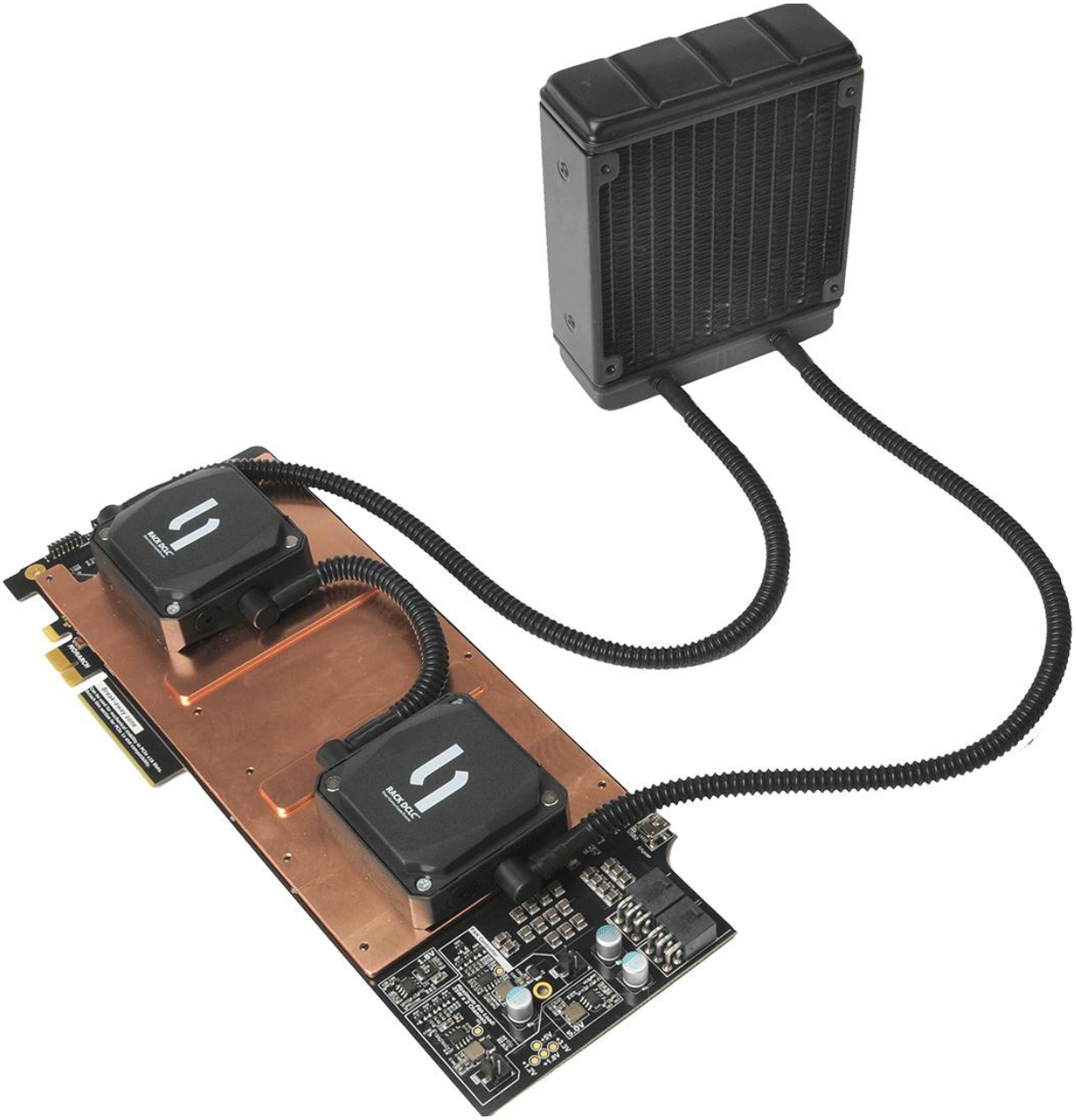

Figure 7 Butterfly Labs Imperial Monarch Mining card & Liquid cooler [16]

## 2.4. Bitcoin Ecosystem

When using Bitcoin, a user stores their coins in a Bitcoin wallet. Wallets are software created by various companies including the Bitcoin Foundation, who produce the Bitcoin QT wallet software. Some wallets offer additional security by encrypting the contents of the wallet in an effort to keep your Bitcoin safe. There are also online wallets where users can store their Bitcoin without having to download any additional software, or in the case of some wallets, the entire Block chain. A user simply needs to create an account and the online wallet will give them an address that they can send their Bitcoin to and it will then be added to their account.

This can be useful for anyone on a company network or users that are behind a firewall. Likewise these services are useful for people on platforms that restrict Bitcoin apps such as Apple's IOS. There are also many wallets available for Android based smartphones. These mobile based wallets allow users to spend Bitcoin at various locations, rather than just online. Another key part of the Bitcoin ecosystem are the Bitcoin exchanges. These exchanges are websites where users can go and trade Bitcoin with one another for Fiat currency. It is also possible to purchase Bitcoin from these exchanges and transfer it to your online or local wallet. Some popular exchanges are Coinbase, based in California and offer online wallets as well as an exchange, and Bitstamp, based in Slovenia. Mt.Gox was the largest exchange worldwide until in early 2014 when the exchange shut down with 200,000 Bitcoin locked in user's accounts.

Both online and local wallets contain the user's public addresses that can be used to send coins to a user on the network and the corresponding private key. These addresses are created from a cryptographic key pair by processing it in several ways. This processing is carried out by the users Bitcoin wallet either on a local machine or an online wallet provider. First of all a Private Key is chosen either by the user or randomly generated by the machine. Then the 65 byte public key is generated. The public key is made up of three parts, a static header and two 32 bit integers referred to as X and Y. This public key is then hashed twice first using SHA256 and then using RIPEMD. Then the Bitcoin version ID is appended to the front of the hash in most cases it is 0x00 for the main network, this is translated in the final address encoding to 1 which is also why many Bitcoin addresses start with 1.

This extended address is then hashed twice more, this time it is done using SHA256 for both instances of it. The output of this hashing stage is then used to create the check sum of the address. This is used to check the integrity of the address to ensure it has not been tampered with or entered incorrectly. The first four bytes are used as the checksum and are appended to the end of the extended address. This is the complete address and just needs to be put through a base 58 encoder to convert it to a Bitcoin network address. [17]

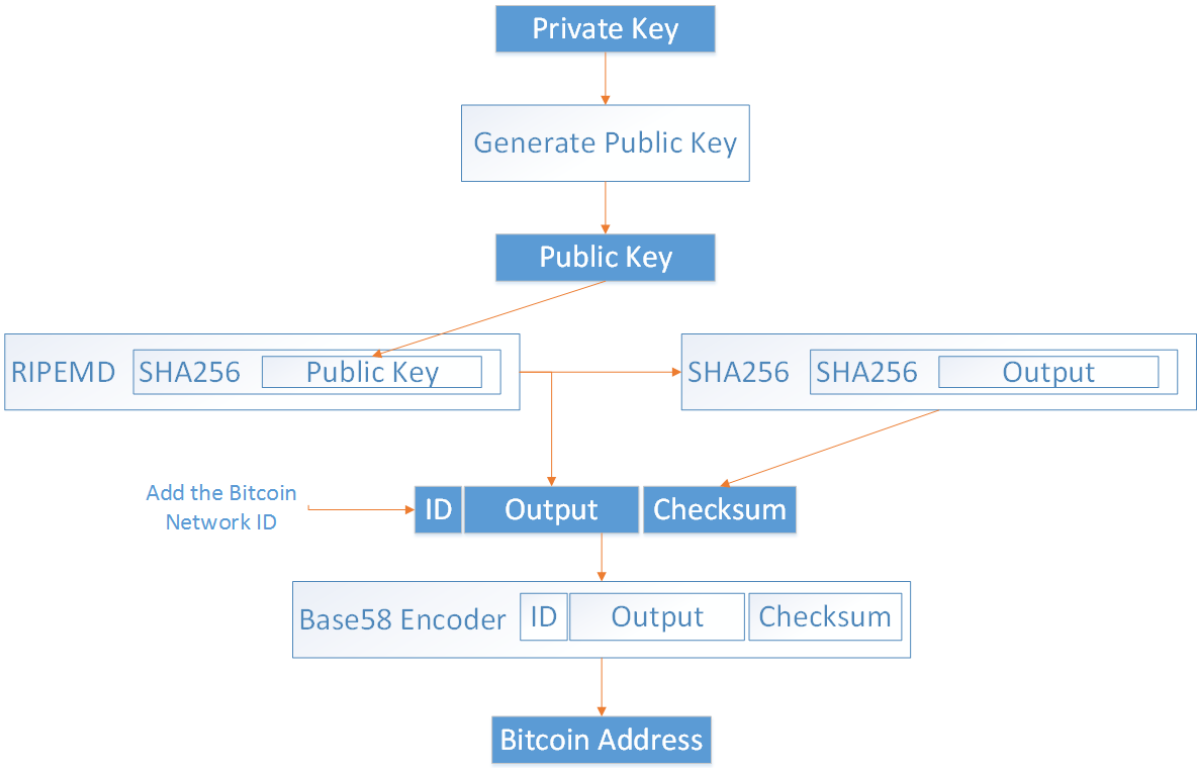


*Figure 8 Bitcoin Address Generation from a Public/Private Key pair*

These keys are inherently secure as the process of stealing another person's coins is very complex and would take a large but indeterminate amount of time. The first step would be to acquire the person's public key from the Block chain. The attacker would then need to get two more pieces of information first is the victim's private key and the second is the hash of the transaction that sent the victim the coins. The private key can be obtained in two ways, the first would be to use a brute force attack and try every key until the attacker has the corresponding key for the public key. However due to the size of a Bitcoin public key this would mean that there are $2^{256}$ possible keys for the attacker to try. The second method of obtaining the victims key is to either remotely or physically access their computer and copy the files. This is assuming however that the user does not have an encrypted wallet. Once the attacker has all of these pieces of information only then would they be able to impersonate the victim and send their coins to the attackers address. Alternatively the reversal of the one way function that creates the public key would be a catastrophic event for the Bitcoin network. Since the one way function is simply a set of mathematical operations it is likely that there exists another function that produces the reverse. If this function is found then every users Bitcoin would be at risk of being stolen. In 2004 Xiaoyun Wang and Hongbo Yu demonstrated that using a differential attack that it was possible to break the MD5 cryptographic algorithm. [18] Furthermore in 2010

CERT, the Software Engineering Institute of the American government, declared that MD5 was now considered completely broken and recommended against its usage. [19] Should something like this eventually happen to one of the hash algorithms used by Bitcoin it could have catastrophic consequences for the network.

## 2.5. The 51% attack

The above demonstrates how it is very difficult to obtain and spend another users coins, and I previously mentioned how the network prevents double spend attacks, however what would happen if there were malicious nodes in the Bitcoin network? In Satoshi's original paper he describes this very situation. In the paper he describes how the malicious node would need to sign transactions faster than the rest of the network. The 51% attack involves a malicious node creating a branched version of the Block chain which they must make longer than the original Block chain. First the node would need to change the transaction in favour of the malicious party and recompute the correct hash. Then the malicious node would need to try to sign the block as fast as it could by finding the collision before any other node in the network. This would work because the Bitcoin network automatically accepts the longest chain as the correct Block chain. Alternatively the malicious node could attempt to modify a transaction that happened in a previous block to favour the malicious party. This would however mean that the malicious node would need to catch up and move ahead of the original chain as the network will accept the longest chain.

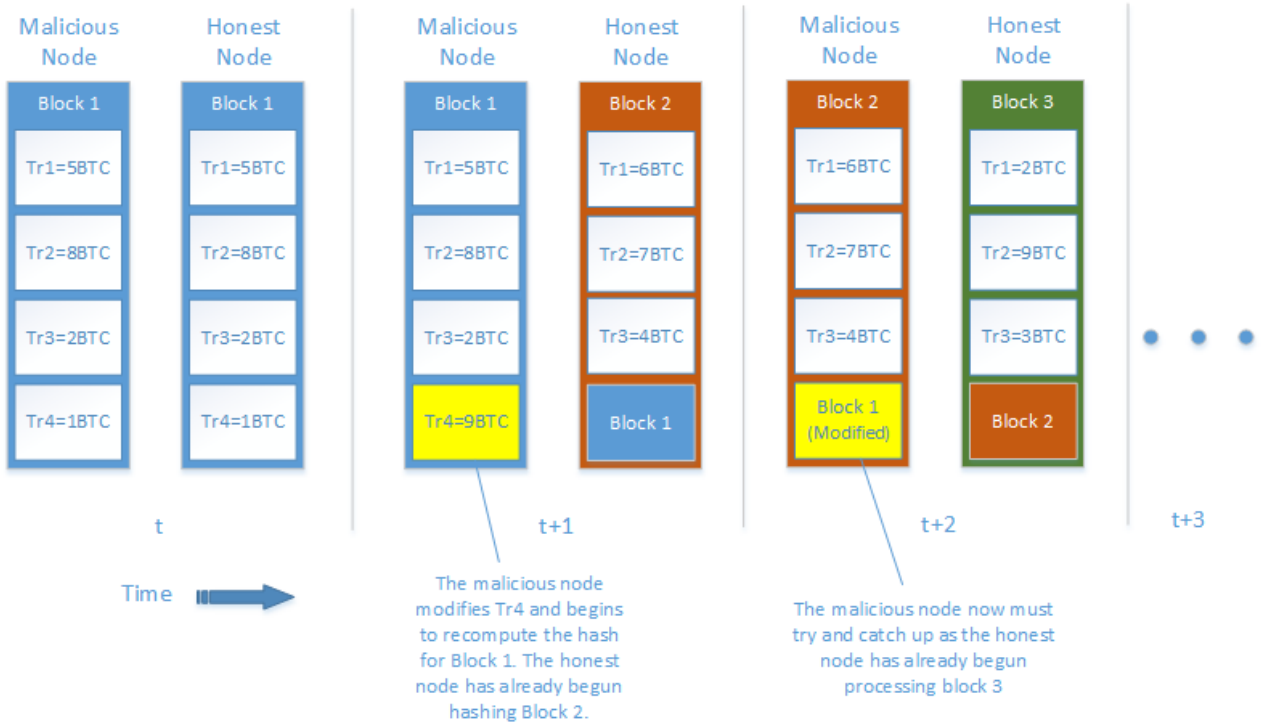


*Figure 9 Double Spend Attack*

In the above example it is shown how the attacker must try and catch up with the honest node. The honest node continues on with the unmodified Block chain so once the malicious node has completed modifying the transaction to its benefit it is already behind the honest node. The only way for the malicious node to catch up is if it possesses more computing power than the honest node. This attack only becomes plausible once the malicious nodes make up a large portion of the network and hence is known as the 51% attack in Satoshi's original paper. However in more recent research it has been shown that due to the mining process relying on probability a group of malicious nodes that make up 10-25% of the network would pose a serious threat to the network and could potentially double spend any of their coins. [20] If the malicious node can create a modified Block chain longer than the main chain then their modified transaction will be accepted by the network. This attack can only reverse or reduce the amount involved in transactions belonging to the malicious node. It is not able to create new coins or take coins from another users address because honest node will not accept them as valid inputs.

Satoshi notes in his paper that this problem can be categorised as a Binomial Random Walk. In this case there are two conditions: a win condition where the honest nodes calculate a block and extends their lead by one, and a failure condition where the dishonest nodes calculate a

block and decrease the honest nodes lead. Satoshi makes the analogy of a gambler beginning in a deficit, playing a game with an unlimited amount of credit in an effort to break even. He provides pseudo code for this along with the output results. It can be seen in the results that there is an exponential drop off in the probability of the malicious node catching up as the number of blocks the honest nodes create increases. Figure 10 shows pseudo code taken from Satoshi's paper that is used to calculate the probability of an attacker successfully carrying out a 51% attack from being Z blocks behind the honest nodes.

```
#include <math.h>
double AttackerSuccessProbability(double q, int z)
{
    double p = 1.0 - q;
    double lambda = z * (q / p);
    double sum = 1.0;
    int i, k;
    for (k = 0; k <= z; k++)
    {
        double poisson = exp(-lambda);
        for (i = 1; i <= k; i++)
            poisson *= lambda / i;
        sum -= poisson * (1 - pow(q / p, z - k));
    }
    return sum;
}
```

```
Results:

q=0.1
z=0     P=1.0000000
z=1     P=0.2045873
z=2     P=0.0509779
z=3     P=0.0131722
z=4     P=0.0034552
z=5     P=0.0009137
z=6     P=0.0002428
z=7     P=0.0000647
z=8     P=0.0000173
z=9     P=0.0000046
z=10    P=0.0000012
```

*Figure 10 Pseudo code demonstrating the probability of an attacker succeeding* [1]

## 2.6. Comparison with other crypto currencies

There are many other crypto currencies available instead of Bitcoin. The next largest after Bitcoin is a crypto currency called Litecoin. Similar to Bitcoin, Litecoin uses a distributed peer to peer network to authorise its transactions. What sets the two crypto currencies apart are the following: Market cap, Hashing algorithm and the time it takes to calculate a block. Bitcoin has a built in cap that is set at 21,000,000 Bitcoins. This means once 21 Million Bitcoins have been mined, whenever a block is solved there will be no reward of Bitcoins. Instead the node that solves it will only get the transaction fees associated with the block. Litecoin however has a cap of 84,000,000 coins. [21] The hashing algorithm used in Litecoin is called Scrypt. It was created in 2009 by Colin Percival and is designed to be a more memory intensive hashing function. [22]

### 2.6.1. Scrypt Algorithm

Scrypt is a password based key derivation function that was designed to be computationally expensive to run and it was initially used for encryption and authentication. Scrypt works by creating a very large vector using a pseudo random number generation function. This vector is

then accessed randomly and mixed using a mixing function and a hash function with the input data. The idea of the large vector is that it must be kept in memory at all times since only certain parts of it will be used to generate the key. The parts used will be different every time hence the whole vector must be kept in memory. This causes the computation to be expensive for a few reasons. [23] Firstly the actual cost of memory is very expensive meaning the initial cost of creating a system dedicated to generating these hashes is much higher. This is advantageous as it prevents highly parallel approaches from being designed as easily due to the higher proportion of memory required. Secondly the latency between the processor and the memory helps to increase the time taken for each operation. Since it is an iterative algorithm with more than several hundred iterations this will also add to the time complexity of the algorithm.

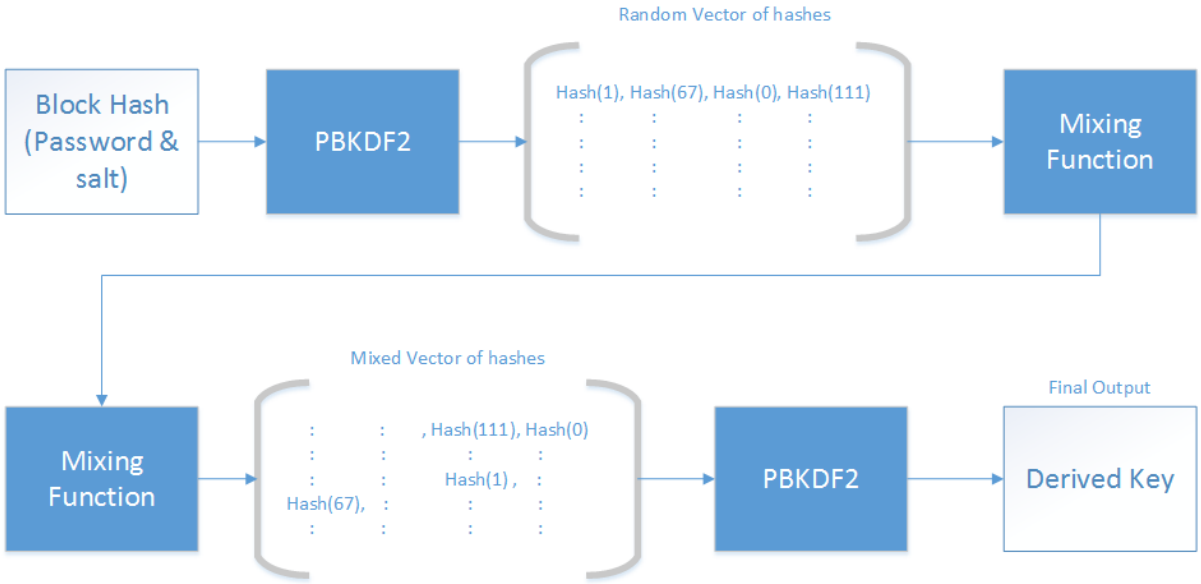


*Figure 11 Scrypt Algorithm*

To generate its pseudo random vectors Scrypt uses a Password Based Key Derivation Function called PBKDF2 and any random number generation function. PBKDF2 uses the pseudo random number generator to create many keys which are then used as the elements in the random vector. This function takes in a password, a salt and an iteration counter among other things to generate the numbers. In the case of this algorithm the salt is a randomly generated number which is concatenated to the end of the password. The aim of a salt in a cryptographic function is to increase the time and difficulty required to crack the initial password by adding this pseudo random number to the end of it. In Litecoin the password and salt are the hash of the block header, similar to a Bitcoin hash, however the Bitcoin block header is simply hashed using SHA-256 as opposed to Litecoin which is processed much more heavily. The mixing

function that is used by Scrypt is called SMix. This mixing function applies the SHA-256 hash function to various parts of the vector, several sections at a time. This produces a mixed vector of newly hashed entries, the same length as the original vector. Once mixed the vector is then once again put through the PBKDF2 algorithm. The output of the second pass is of a shorter length and is now the final hash of the block. The block is then checked against the difficulty to see if it meets the criteria of the current block difficulty.

### 2.6.2. Alternate Mining methods

There are two main ways that cryptocurrencies are mined. The first is Proof-of-work, this is what is used by Bitcoin and Litecoin. The second method is known as Proof-of-stake. Proof-of-stake works by only allowing people who hold a specified amount of the currency to authorise transactions. This is intended to prevent the 51% attack that Bitcoin is susceptible to. The argument for proof-of-stake is that for someone to have 51% of the power of the network they need to have 51% of the currency. The only way for someone to achieve this is by buying up the currency thus increasing the price and making it more and more expensive for them to reach the required 51%. Once someone has bought up 51% of the currency it would be completely against their interest to attempt a 51% attack since they would end up with a massive capital loss if they attempted to disrupt the network. An example of a cryptocurrency that uses the Proof-of-stake concept is Peercoin. [24] In Peercoin a miner who owns 5% of the currency in circulation has more power than a miner only holding 1% of the currency. This essentially means that if both miners sign a block but the miner who owns 1% attempts a double spend attack or some other malicious attack, the network will ignore that miners block and accept the block signed by the miner with 5% of the currency. While Litecoin and Peercoin have both been moderately successful their adoption rate has been much lower than Bitcoin's. This is reflected in the price of the currencies while at the time of writing Bitcoin is $450 per coin while Litecoin and Peercoin are $10 and $2 respectively.

## 2.7. Bitcoin controversies

Over the course of the last year Bitcoin has enjoyed an ever growing amount of press coverage. A large amount of this coverage began in April 2013 when the Bitcoin market price soared to over $240 and the dropped overnight down to close to $100. Since then Bitcoin has featured on many news broadcasts, panel shows and discussions. Not all of this coverage was positive and I will discuss some of the negative press and the repercussions it has had on the network.

### 2.7.1. Bitcoin theft

There was a paper written in 2011 by two UCD researchers named Fergal Reid and Martin Harrigan that analyses privacy in the Bitcoin Block chain and analyses one of the largest Bitcoin thefts at the time. [2] They begin their paper by discussing how Bitcoin operates. They then move on to discuss two abstractions upon the Bitcoin network. One they call the Transaction Network and another called the User network. The transaction network is described as mapping the flow of Bitcoin through the network by examining the transactions. The user network is described as being the interactions between different users in the form of transactions.

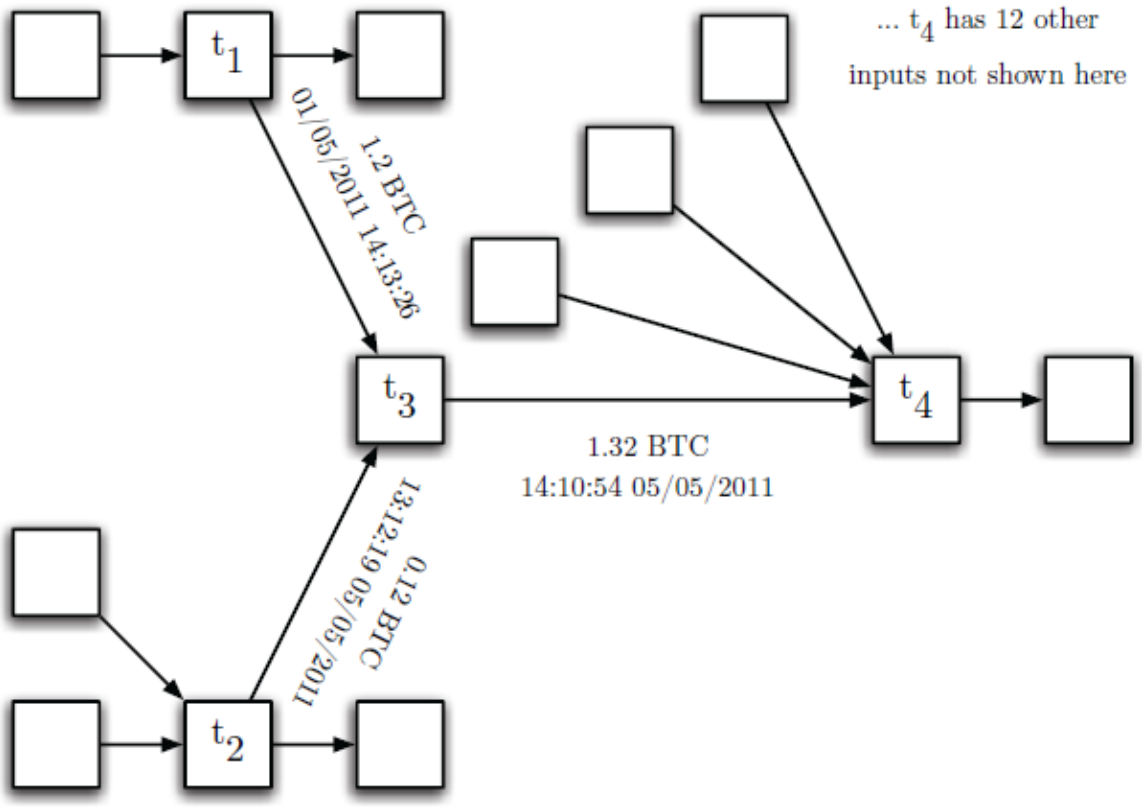


*Figure 12 Transaction Network Graph* [2]

It can be seen in the above diagram taken from the Reid and Harrigan paper what is meant by the Transaction and User network. The above is a graph of the Transaction Network showing Bitcoin moving in four transactions to a final address on the right of the graph. Looking at $t_1$ we can see an empty box leading into it representing the input address being used in the transaction. The graph edge that leads horizontally out of $t_1$ to another empty box represents

some Bitcoin being sent to a shadow address belonging to the original owner of the starting box. The diagonal edge that leads to the box labelled $t_3$ represents the output address of the transaction. This is the Bitcoin being transferred from the original owner to the owner of the address used in $t_3$. The authors then move on to perform a case review of a Bitcoin theft and they show where the coins go after the theft using their Transaction and User network graphs. In the grey box on the left it can be seen where the coins are transferred from the Victims mining pool account, represented by the yellow circle, and into the thief's address, represented by the orange circle. The diagram then shows the coins being scattered over dozens of different addresses as the thief attempted to cover his tracks by confusing anyone following the trail of the Bitcoins. It can be seen how most of the Bitcoins flow out to dozens of different addresses before all being deposited in an address, represented by the large red circle, known to belong to an online wallet provider known as MyBitcoin, which has since ceased trading. This shows us how a transaction network graph can be used to track a user's actions through the Block chain by following the flow of transactions.

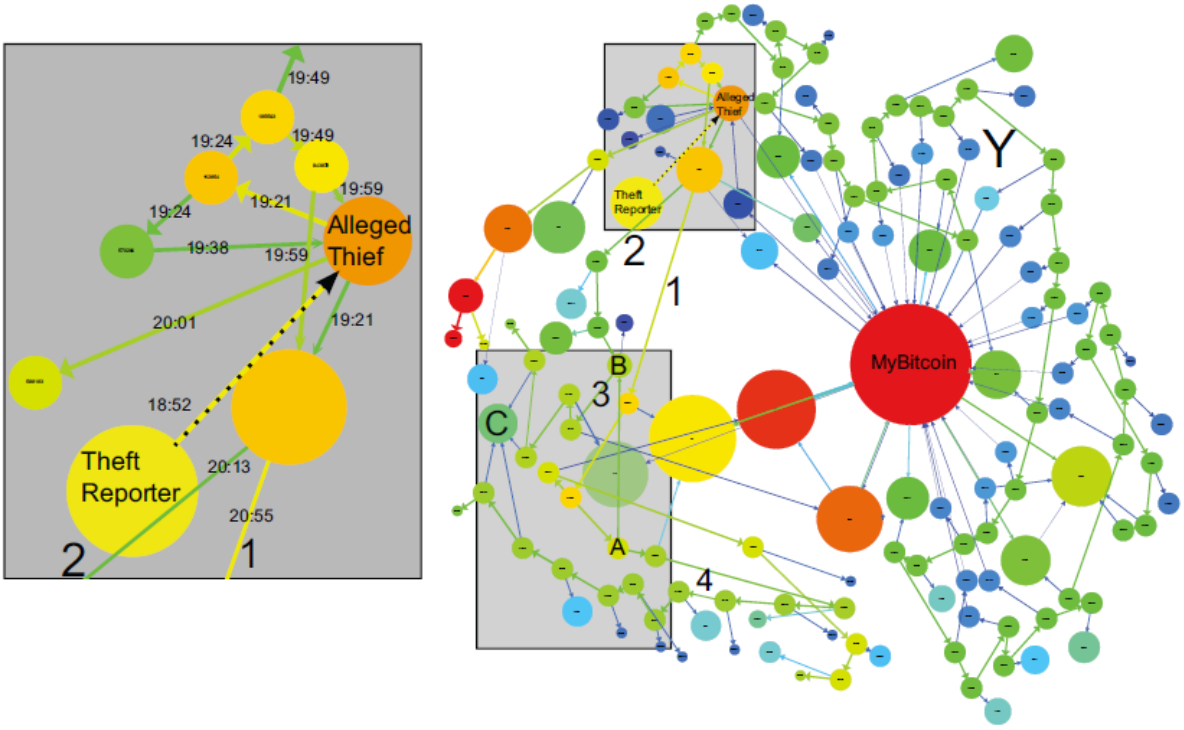


*Figure 13 Transaction network showing a Bitcoin theft* [2]

### 2.7.2. The Silk Road

The Silk Road was created in February 2011 [25] as an online market place for illicit drugs and weapons. The Silk Road used Bitcoin as an anonymous payment method between consumers and various narcotic dealers. The service was hidden behind the TOR anonymous network and registered to the name Dread Pirate Roberts. The TOR network is an anonymous network that allows users to try and hide their IP address. The TOR network does this by using what is known as an onion network. This is where the users request is encrypted before being given to a web service is bounced around hundreds of different TOR nodes in an effort to make it difficult to track a user's browsing history. [26] Anyone can download the TOR browser to take advantage of the TOR networks privacy. Likewise anyone can set up a TOR relay and act as one of the nodes in the TOR network. In October of 2013 the Silk Road was shut down and those connected with it arrested by the FBI. All Bitcoin held by the site was also seized by the FBI. In January of 2013 the FBI announced they were going to auction off some of the coins seized from the Silk Road however they have yet to sell them. [27]

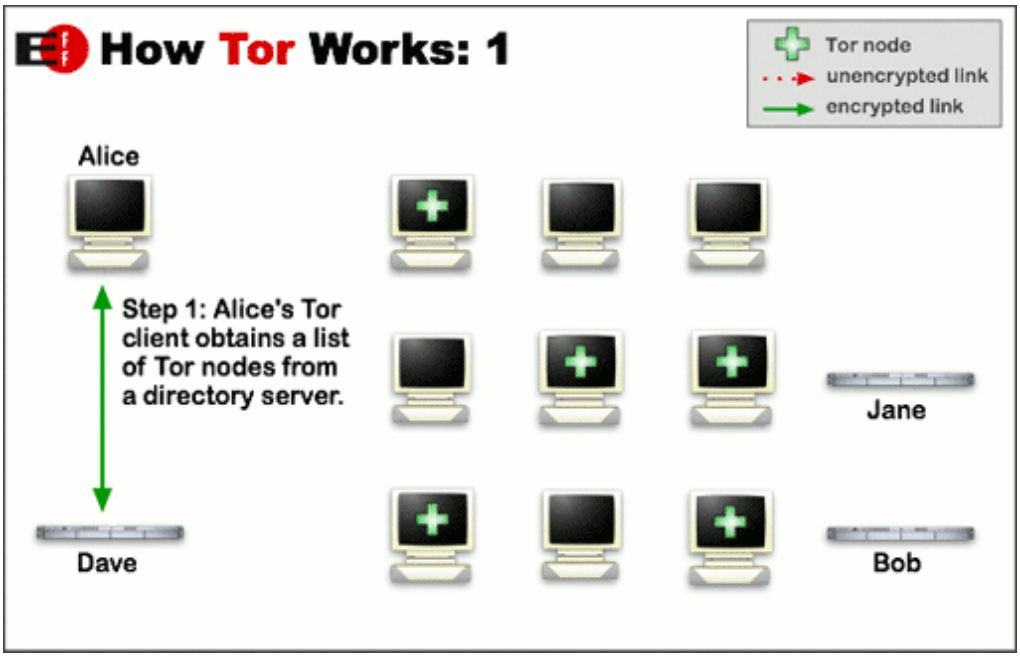


*Figure 14 How TOR works Part one* [28]

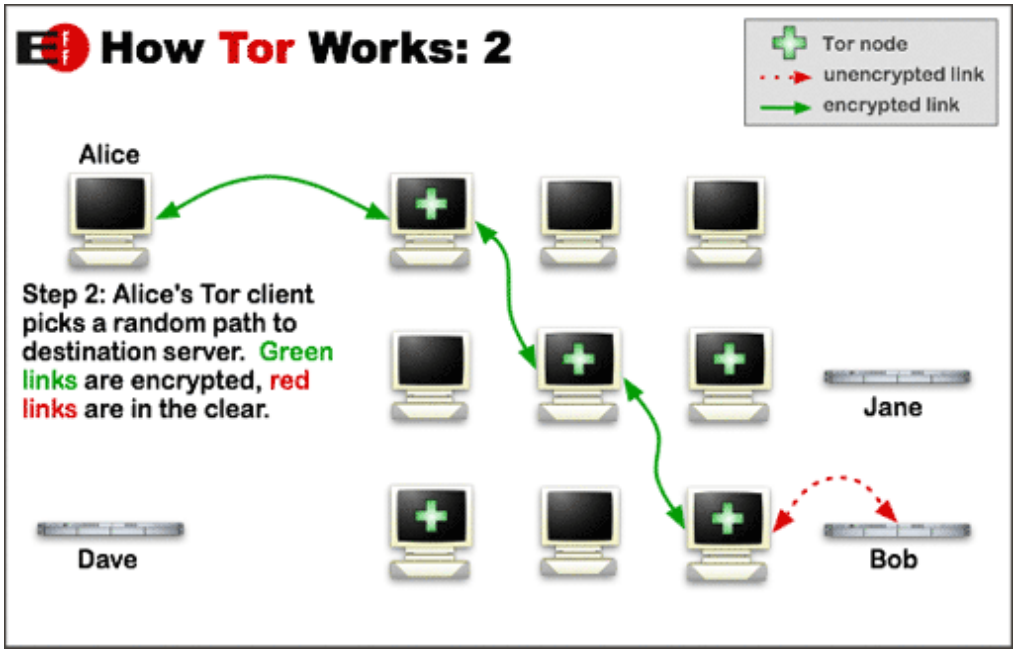


*Figure 15 How TOR works Part two* [28]

### 2.7.3. The Record High

In November of 2013 Bitcoin reached over $1000 dollars for a single coin. [29] This was a record high price and attracted a lot of media attention and speculation on the future of Bitcoin and if widespread adoption was possible. Within a week of breaching the $1000 barrier, the price of Bitcoin fell drastically to just over $700. For several weeks following this there was a lot of uncertainty in the Bitcoin community and it was easily to see this reflected in the price. The price began to wildly fluctuate from as low as $700 dollars to over $900 increasing speculation that Bitcoin was following the economic definition of a bubble [30]. Dr. Jean-Paul Rodrigue outlined the various stages of a bubble in 2008. When compared to the price of Bitcoin over the last year his graph of an economic bubble is remarkably similar. There are many similarities between the two like a clear bear trap, a "new paradigm" peak way above the mean, a bull trap and a resurgence in the price before dropping back towards the mean price. I have marked these areas on a price graph from October 20th 2013 to December 25th 2013 on the chart below.

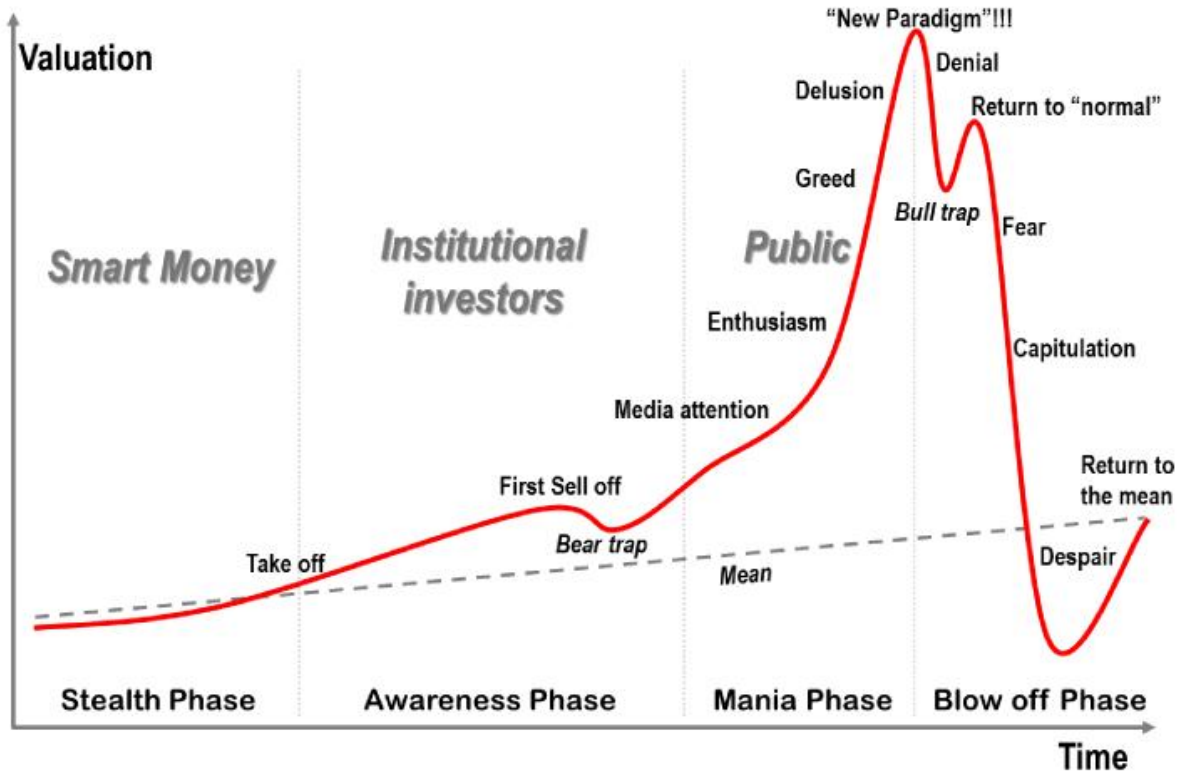


*Figure 16 Dr. Jean-Paul Rodrigue's definition of an economic bubble* [30]

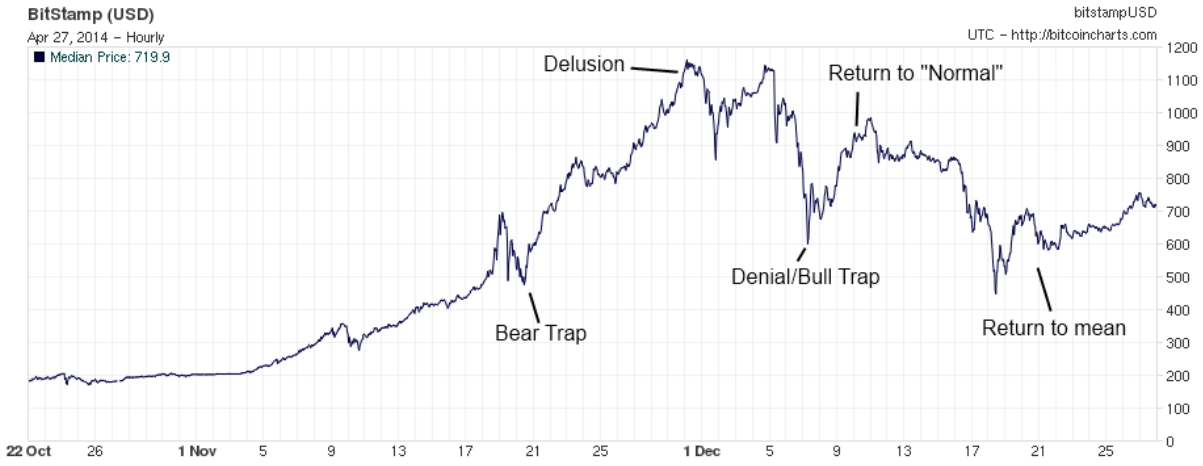


*Figure 17 Bitcoin price chart for October 20th to December 25th 2013*

### 2.7.4. Mt. Gox

At the height of the Bitcoin speculation Mt. Gox was the largest exchange seeing the highest prices and trading volumes of any exchange. This was evident in the price difference between Mt. Gox and other exchanges. Prices could vary by as much as $100. However in late 2013 some users began having issues withdrawing Bitcoin from their accounts and many of them took to forums and social media trying to find out if any others were having issues with their accounts. [31] These issues continued while Mt Gox continued to operate their exchange. In

February of 2014 Mt Gox halted withdrawals on their site in a press release claiming there to be a technical fault that needed the system to be temporarily disabled. [32] This shut down coincided with many other exchanges temporarily closing to patch a bug that allowed for something referred to as "transaction malleability" which I will describe later in my experimentation section. [33] After much speculation Mt. Gox ceased trading completely and shut down their service claiming that 850,000 Bitcoins, valued at roughly $4.9 billion, belonging to its users had gone missing. In late March 200,000 coins were found in an old wallet while filling for bankruptcy. Mt.Gox stated that they were found in an old wallet that was believed to be empty. This address was found after the company had closed its physical office and no effort has been made to return these coins to their rightful owners. [34] I will discuss what happened to Mt.Gox in more detail in my evaluation section where I will attempt to analyse what really happened to their user's Bitcoin.

#### 2.7.5. Apple app store

Apple have made their stance on Bitcoin clear on several occasions. They have removed many apps from the app store ranging from Bitcoin wallets to Bitcoin price tickers. Every time this occurs Apple cites rule 22.1 in their developer terms of service:

> *"Apps must comply with all legal requirements in any location where they are made available to users. It is the developer's obligation to understand and conform to all local laws."* [35]

While many members of the Bitcoin community are outraged by this, like Rob Banagale CEO of Bitcoin start up Gliph [36], one of the companies to have their app removed by Apple. Since there has been no definitive ruling from the US government regarding the legality of Bitcoin and its classification as a currency or an asset, it is unlikely that Apple will allow Bitcoin apps on the app store. [37]

## 2.8. Relevant Technologies

It was important that I chose the technologies that I was going to work with early on in my project so that I was able to get started on writing the necessary code as soon as possible. I identified three main technology choices that needed to be made, the language I would write my programs in, the database I would use and the data format that I would use to get information into my application and database.

### 2.8.1. Programs and scripts

The programs and scripts that I wrote were created in Python. Python is a simple scripting language and is open source. It was created in 1991 and is managed by the Python software foundation. Its current version is Python 3.4 however many users still use Python 2.7 as many do not like the changes made in version 3. Google's app engine only supports Python 2.7. [38] My reasons for choosing Python were that it has a very simple syntax which uses indentation and white space over parenthesis and various line ending symbols. I will demonstrate this in a simple comparison with Java.

Python:

```
def function(a, b):
    print a+b
```

Java:

```
public void function(int a, int b){
    System.out.println(a+b)
}
```

This is a very simple example but it is very easy to see that the Python sample is much shorter and cleaner looking. Both examples do the same thing: take in two values and print out the sum of them however the python version is more powerful as it will work for any type while the Java example only works with integers. Another reason is that Python interpreters are available on many cloud based platforms making it easy to use my scripts and programs on many different services like Amazon Web Service and Google App engine. This would allow for easy scaling of the system whenever it needs extra power, like at times of high load.

### 2.8.2. Database selection

All of my programs and scripts generate a vast amount of data on the Bitcoin Block chain so it is necessary to have a database capable of quickly storing and sorting through the millions of transactions in the Block chain. I looked at using two different technologies, SQL and MongoDB. I chose to use MongoDB for several reasons the first being that MongoDB is a no-SQL database which was created in the early 2000's in an effort to combat the limitations of SQL at the time. Another reason for using MongoDB is its open source and cross platform nature. Similar to Python the platform I choose to run my implementation on is not an issue because of this. The final reason for using MongoDB is its flexibility. Like other no-SQL

databases it is capable of storing objects, documents and key-value pairs in the same table. MongoDB stores its data in JSON documents. These documents are easily interpreted by Python without needing to use any external libraries.

### 2.8.3. APIs and Data formats

Some of my scripts make queries to a website called www.blockchain.info. They have a public developers API available which exposes details on single transactions, single addresses, entire blocks and has many more query options too. The API supplies the data in either plain text or JSON formats. They also provide a data steam that pushes new data to your service as it is available however I did not avail of this. I used the JSON API in my scripts which made for simple parsing and storage of the data due to Python being able to manipulate JSON data in a very simple manner and MongoDB storing its documents in a JSON format already.

JSON is a simple plain text method of representing data. It is an acronym for Java Script Object Notation and its design was based on JavaScript's representation of objects. It was modified slightly from JavaScript syntax so it would be easily readable by both humans and computers. JSON was a simple choice for me as it is already the syntax of MongoDB data stores, it is the same format as the format as the data returned from the www.blockchain.info API and it is easily manipulated in Python.

## 2.9. Reputation systems

Rating systems are computer systems that are designed to correlate data about a user or data set in an effort to produce a score representing their standing in the community. Systems like these became common place on internet market places to help users trust one another. These ratings can be created from empirical data or from user ratings, similar to the systems found on many modern online auction websites. As I will discuss later in my design section my system will use a rating system to allow users to rate transactions they have with one another. One problem with reputation systems is user manipulation. This could range from giving unjustly harsh or overly generous reviews to a user attempting to manipulate their own reputation score by leaving false reviews. Andrew Whitby, Audun Jøsang and Jadwiga Indulska outline a method in their paper filtering out unfair ratings in Bayesian reputation systems to prevent unfair reviews from altering the ratings [39].

In this paper the authors propose a method for removing unfair reviews from a reputation system. As I will discuss later in my design section a rating system would be a useful addition to the system I propose. The authors begin the paper by first describing the basic concept behind a rating system. They describe it as:

> *"to let parties rate each other's performance during interactions. The aggregated ratings about a given party are used to derive a reputation score, which can assist other parties in deciding whether or not to transact with that party"*

The authors then outline two different mechanisms for removing unfair ratings called Endogenous and Exogenous Discounting of Unfair Ratings. Endogenous discounting involves using statistical methods to remove potentially unfair ratings. These use a statistical model to apply lower weights to the unfair ratings. Exogenous discounting involves using extra data such as the reputation of the rater to apply a weight to the rating given. The authors close the section by saying their system falls into the first category and does not rely on external data. In the second section of the paper the authors describe the Bayesian reputation system that their system builds on. These systems use Probability Density Functions, PDF, to create the ratings for the user. When a new rating is made a new PDF is made and combined with the existing one. This ends up updating the users rating. The authors then give a mathematical definition for the beta function of a PDF. This beta function is what is used to update the PDF. The authors then give an example of a neutral rating of one positive and one negative rating shown below on the left, and an example of a user with seven positive ratings and one negative ratings shown below on the right. The authors also talk about how rating age and newer ratings are more influential than older ratings.

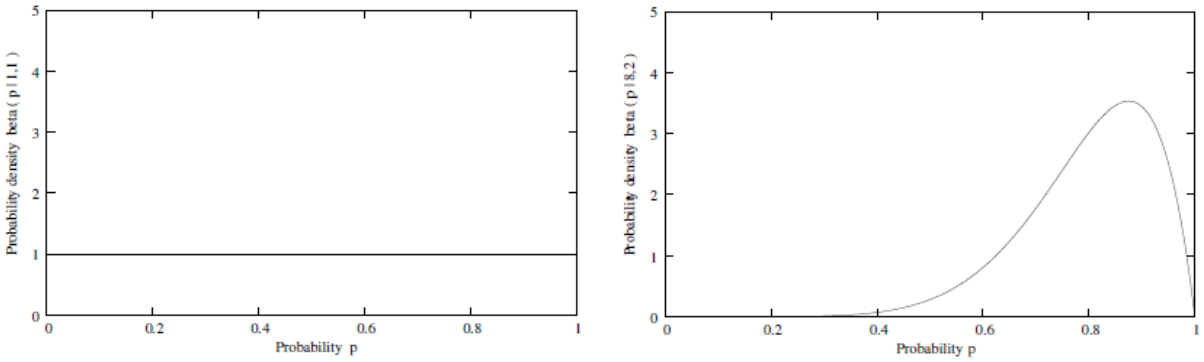


*Figure 18 Example Probability Density Functions* [39]

The authors then move on to the problem of unfair ratings. They define two types of unfair ratings that they want their system to remove. The first is unjustly negative ratings and the second are overly positive ratings. They decide to do this by filtering when a rating is being added to a user's over all PDF. The authors describe their approach as filtering out reviews that do not fit within a given quantile range. By this they mean that the 1% quantile is the values that are less than 1% of the data set. One of the examples they give is to use only values between the 1% quantile and the 99% quantile. This gave a user rating that was deemed to be fairer based upon the correlation of all ratings for that user. This meant that user ratings were only strongly affected when they began to receive several very high or very low ratings.

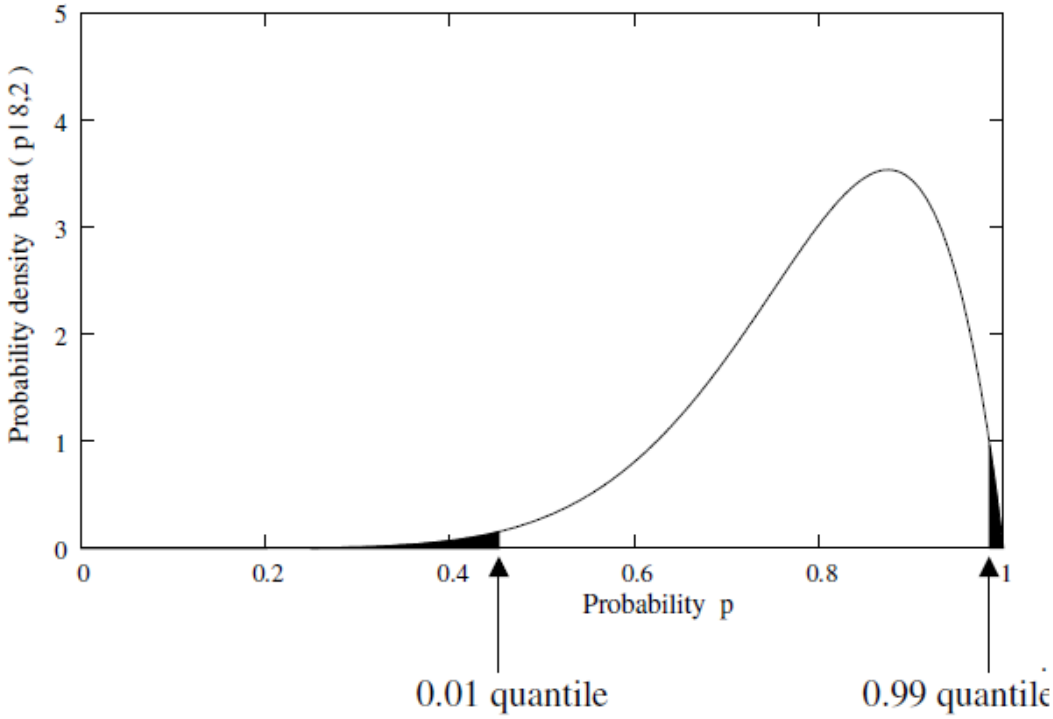


*Figure 19 Filtered Probability Density Function* [39]

The authors discuss how they simulated a market place and that had buyers and sellers each with their own reputations. They tested how these reputations were affected by new ratings and how these ratings were influencing buyer's actions. They simulated unfair ratings with and without filtering and it was clear that the filtering provided a more consistent rating for the seller while without filtering the rating would wildly fluctuate.

This system is capable of preventing unfair reviews from affecting the overall rating for a user. Later in my design section I will use this design and advance upon it to prevent users from manipulating their own score or another user's score by using fake reviews.

## 2.10. Taxation and regulation

Each year in the US citizens are required to fill out a tax return form and return it to the Internal Revenue Service. This form is used to calculate income tax as well as tax credits and refunds. On the 18th of March 2014 the IRS became the first government organisation worldwide to tax any commercial activity in Bitcoin. The IRS announced their plans in a press release where they announced that Bitcoin would be taxed as a property and not a currency. [40] Owners of Bitcoin in the US are required to declare the date they received the Bitcoin how much Bitcoin they received and the fair market price of the Bitcoin at the time. The IRS define the fair market price as being the price of Bitcoin as decided by an exchange based upon supply and demand. This means users would have to track back through price charts and their own transaction logs and calculate what tax is payable. A user has to declare each time they received Bitcoin either as a payment for some goods or services, or bought some from an exchange. This works similarly to how someone would go about declaring stocks that they have bought

The difficulty with this is that due to the anonymous nature of the Block chain it is very difficult to ensure that every user has declared the Bitcoin that they own. It would be possible to audit someone and notice unusual payments and lodgements in their bank account but first working out who to audit would be very difficult. Since Bitcoin addresses are arbitrary strings of base 58 characters it is impossible to directly link them to a user. They do not contain any personal information and there is no central register of users and their addresses. The only way to be certain that a user owns an address is by getting them to send some Bitcoin from the address. Due to this fact it would be impossible for the IRS to know several things about American Bitcoin users. They would be unable to know how many addresses a citizen owns. They would be unable to tell what country the owner of any given address is from and they would not know how much Bitcoin a person owns. This makes it very difficult for the IRS to guarantee that people will put accurate declarations on their tax returns. It would also make it difficult for the IRS to audit a person should they have doubts about the information they have provided regarding the quantity of cryptocurrency they own. Using the system that I will outline in the next chapter it would be possible for the IRS, or any other financial regulator, to use one address belonging to a citizen to calculate their total Bitcoin wealth.

In this chapter I discussed how Bitcoin operates. I talked about the Bitcoin community of miners and the ecosystem needed to support them. I analysed Bitcoin when compared to other cryptocurrencies. I went on to mention several events in the history of Bitcoin that received a lot of media attention. I finished the chapter by discussing the technology choices for my

project, the usage of reputation systems in an online environment and Bitcoin regulation and taxation. In the next section I will discuss the implementation of the algorithms and scripts that I have created. I will also outline the design of a system that can build upon these algorithms and provide a service to the Bitcoin community. Lastly I will talk about any additional work that could be continued on from my design and implementation.

# 3. Design

## 3.1. Overview

In this section I will discuss the technologies, design and implementation of my Bitcoin Block chain tools. I will lay out the design of a system that uses these tools to provide a service to be used by financial regulators to monitor the Block chain or the Bitcoin community to check the trustworthiness of an unknown address. I will then discuss the implementation of these tools.

The tools I will discuss are my web crawler, my Block chain parsing tool and my group creation tool. These three tools provide the basis for the information that the designed system would use. The designed system outlined in the second part of this chapter is intended to help users gain some confidence when it comes to dealing with anonymous Bitcoin addresses. I aim to achieve this in two ways. The first is to create groups of addresses known to belong to one user. The second part is to provide a rating for the trustworthiness of that user by examining both ratings from other users and by looking for malicious transactions in their spending history.

This rating will then give users an insight into who owns the anonymous address they are about to send their coins to. They will be able to see the users history, their estimated worth and they will be able to see the ratings left by other users.

## 3.2. System Design

This is the definition of a system that would be capable of parsing the Blockchain and serving up data that has been deduced. This data is deduced by examining the Blockchain and using some properties of a Bitcoin transaction. These properties allow us to make some assumptions within the data and from them we can create links between addresses and user. The system I designed is intended to be used by both everyday users of the Bitcoin network and financial regulators. The system can be used to get more information about an address to see if it is trustworthy or not. It can be used to rate a transaction with another user which is then used to give the communities opinion on how reliable the owner of the address is. The system can get an estimation of a user's overall value which could be useful for taxation purposes and it can be used to monitor transactions or addresses.

### 3.2.1. Cloud technologies and Scaling

To scale a system to provide a service to the entire Bitcoin community it would be necessary to put a lot of computation power behind such a system. To keep this cost effective a cloud system like Amazon Web Services (AWS) should be utilized. Using a system like AWS it would allow for elastic expansion of the system as the load increases either from an increase in users or a need to process new data to add it to the system. By using Amazons Elastic compute cloud (EC2) the system would be able to add new instances as necessary thanks to the load balancers that can be set up by the user. When the system reaches a certain level of traffic the load balancers can kick in and start up several more instances of the system so that users don't see a slow down with high traffic volumes. More instances could also be started to process the data as new blocks are created by the Block chain.

Since a new block is created once every ten minutes, according to Satoshi's original specification, it would be unnecessary to have the data processing system up and running the entire time as this would use up resources and increase the cost of maintaining the system. Instead it would be more beneficial to have one instance running waiting for a new block. Once a new block has been detected this instance could then launch more instances as necessary to process the block. Once each instance is finished they could then be shut down to save money.

Amazon has a NoSQL database that can be used with any applications running in EC2. This technology is known as DynamoDB and has many features such as atomic actions, automated storage scaling to spread a single table over several servers to help with load balancing and the advantage of having your data replicated to three different locations to add a level of fault tolerance to your application. [41] Using DynamoDB the instances working on processing the Block chain could write the new information to the database while the front end is used to serve this information to the users of the system.

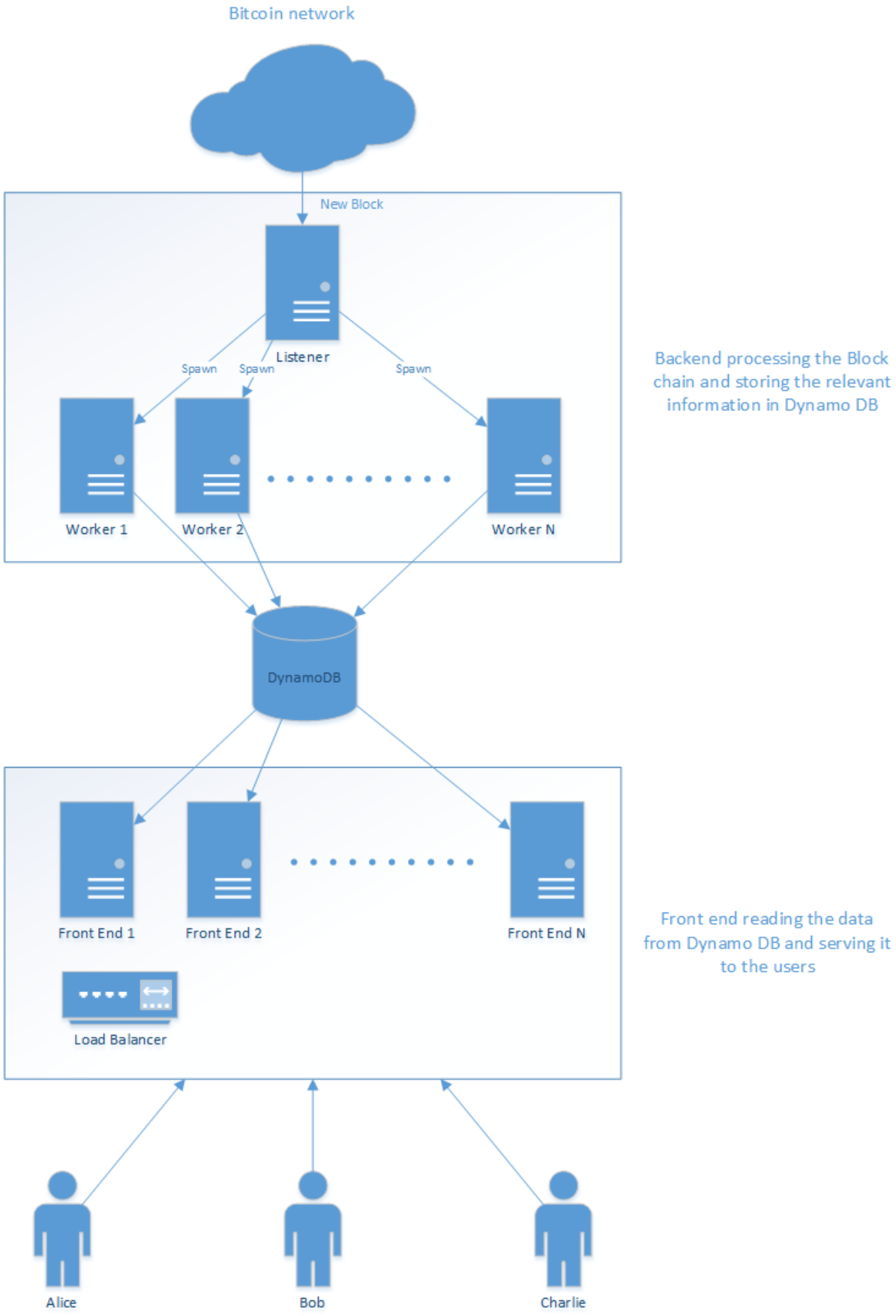


*Figure 20 Overall system design*

### 3.2.2. Supplying information

The system would allow users to view information about any given address. Some of the information available in the system would be the owner of the address if it's known, the user rating of the owner or the address, a list of all transactions it has been used in. It would also show any unusual transactions that have occurred in the Block chain recently and it will allow a user to rate a transaction that they have taken part in. An unusual transaction could be a huge sum of Bitcoin being transferred, or an attempted double spend attack. Patterns like this could be examined by the worker nodes as they are processing the incoming transactions in the Block chain.

### 3.2.3. Authenticating users

One of the key features of the system I am designing would be the ability for a user to rate a transaction that they have taken part in. This would help with trust in the Bitcoin network as currently when a user needs to send Bitcoin to someone, they are dealing with a string of seemingly random characters as the address. This makes it difficult for the user to know exactly who they are dealing with. By entering the address into the system the user would be able to check that the owner is who they say they are, check previous transactions they have carried out and most importantly see how other users rated them.

It is necessary that user ratings are controlled to prevent people from forging a good reputation for themselves. It would also be needed to prevent competitors from leaving negative feedback for each other in an effort to make their own product or service seem better. I decided that some form of user authentication would be necessary. I chose to not use the usual username and password approach for several reasons. The first being the additional storage necessary to save each users account information and settings. This would be an additional and potentially unnecessary cost on top of the existing maintenance of the system. The second reason is securing this data. Due to the sensitive nature of usernames and passwords this data needs to be stored in a very secure way. An optimal way is to not store the username and passwords themselves and just store a hash of them however this could potentially take up even more room than storing the plain text due the varying lengths from different hash outputs.

Instead I would use a challenge based system. This system would be designed to utilize the already existing Bitcoin public and private key pairs as a form of securing transactions and ensuring that the user is who they claim to be. This can first be done by the user finding the transaction they wish to rate on the system and stating that they are one of the involved parties

in the transaction. The system would then generate a challenge for the user based on their claim to one of the addresses.

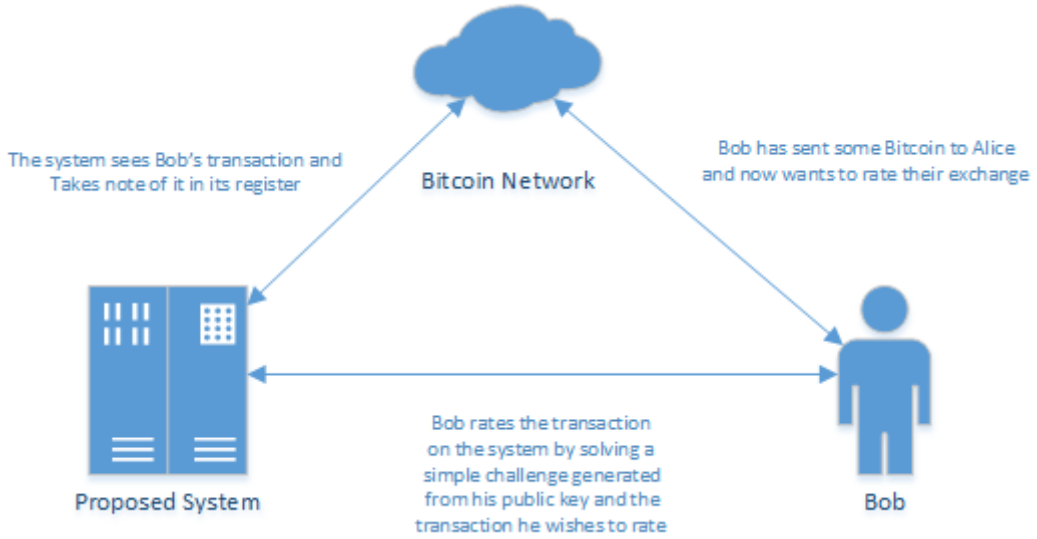


*Figure 21 System Blockchain Integration*

Continuing the example from the above figure, if Bob wants to rate his transaction with Alice he must first tell the system the he is the owner of the input address and wishes to rate the transaction. The system will then generate a key pair. One public, $PubK_{1,}$ which it will encrypt with Bobs public key, $PubK_{Bob,}$ for his address. The second key being the private key, $PrivK_1$ which will be kept by the server for the final step of the process. The public key of this pair is encrypted with the public key associated with Bobs address, $Encrypt^{PubKbob}(PubK_1)=Msg1$, meaning the only way for anyone to access this key is by using the private key associated with Bobs address, $PrivK_{Bob}$, to decrypt it. Once it is has been encrypted it is then sent to Bob where he must begin the first step of proving he is who he claims to be. He must decrypt the message sent to him, using his private key $Decrypt^{PrivKbob}(Msg1)=PubK_{1,}$ getting the system's temporary public key. Then once he has decided on what his rating for the transaction will be, he uses the system's temporary key to encrypt his rating which is then sent back to the system. $Encrypt^{PubK1}(Rating)=Msg2$. The system can now attempt to decrypt the message received from Bob. $Decrypt^{PrivK1}$ (Msg2)=Rating. If it decrypts correctly and it contains a valid rating then the system knows that Bob was the genuine owner of the address in question. His rating for Alice is then added to the other ratings in the system for Alice.

If the system does not know who Alice is and has simply found the group of her addresses but does not have a username for her then Bob will be able to submit a temporary name. These temporary names would be used by the system to identify Alice until Alice claims an address belonging to her and chooses the name or pseudonym that she would rather use. Once Alice claims one of her addresses the system will then find the group containing the address that she claimed. This group will then be marked as belonging to Alice without her needing to claim each individual address. I will discuss the creation of these groups later in the implementation section of this chapter.

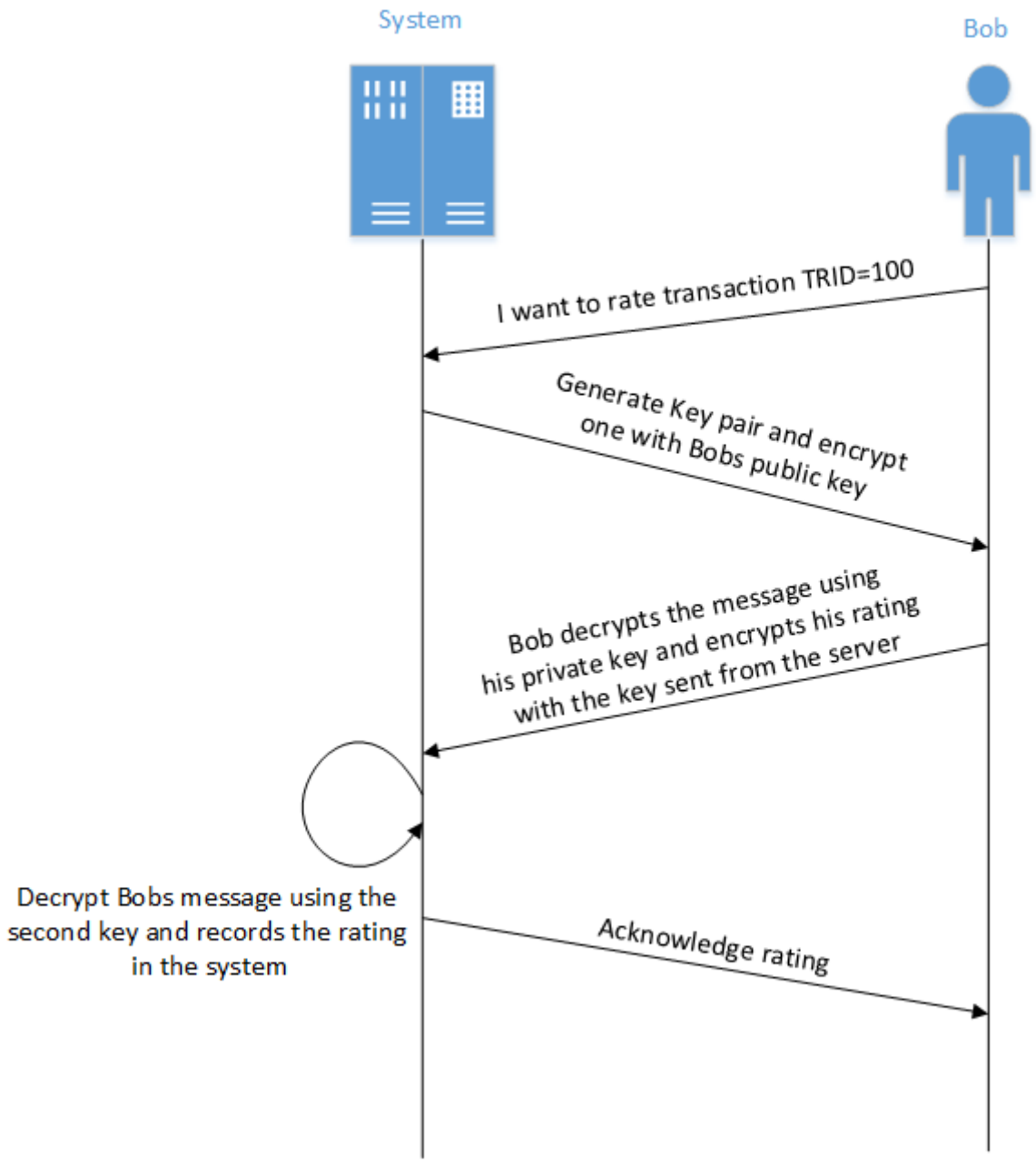


*Figure 22 Authentication Flow*

### 3.2.4. Rating System

As I described earlier in my state of the art section, Bayesian reputation systems are often used to give an overall rating for a user of a system and are also filtered to prevent reputations being tarnished by overly generous or overly harsh ratings. Using the above system to authenticate a user using only their Bitcoin address, it is possible for the system to accept ratings and know that they are genuine ratings coming from the correct user. From there these ratings can be fed into the beta distribution function where the results can then be filtered as described in the section in my state of the art.

In that section I mentioned that these systems are able to filter out unfair reviews, however they are not able to prevent users from manipulating their own reputation by adding false reviews. By using my authentication system it would ensure that users would have to have actually sent some Bitcoin to an address before they can rate it. It would also be important to monitor what transactions have been rated to prevent multiple ratings being entered for the one transaction. By doing it this way the system would end up costing people money to manipulate their own reputations. First they would have to buy or build a bot net to make lots of transactions so that they can be rated. On top of this bot net they would also end up paying miners fees costing them more money for each transaction they make. Another way to increase this cost and act as a deterrent is to store the ratings on a user to user basis. This means that only one of Alice's ratings for Bob will be stored in the system and any subsequent ratings will simply overwrite her previous rating. If Alice wanted to change her rating for Bob she could simply make another transaction with him, however it would prevent Alice from corrupting Bob's rating with several unjustly negative ratings or overly positive ratings.

## 3.3. Implementation of the tools

As part of my implementation I wrote the scripts that would be run on the worker nodes. There are three main scripts which I will discuss their algorithms in detail. These scripts make up the back end of the system I have already outlined. I will discuss in more detail the flow of data between the scripts within the back end of the system. I also will outline my database schema for the partial and complete data and I will note what changes would need to be made to my implemented database for it to be used with the design I have already described.

### 3.3.1. Database Design

The first and most important part of my design was my database schema. Due to the nature of my implementation which has several scripts producing and modifying shared data it was important that I had a consistent data model to work from so that one script would not cause another to fail by modifying the schema of the data it was working with. First I had to work out the flow of data from various sources and through my scripts and database.


*Figure 23 Data Flow Through The Application*

I decided to work from the bottom up while working out the flow of data. Blue squares represent database tables, Green squares represent scripts manipulating data and Red squares represent sources of data. It can be seen here that my crawler script and Block chain parser script both add to the User and Key database. This table stores a username, email or some other identifier and a Bitcoin address known to belong to that user. The Transactions table stores

every transaction from the Bitcoin Block chain once they have been stripped down by the Block chain parser script. These two tables are then used by the Key Groups script which produces a group of addresses that belong to one user. The crawler is designed to find Bitcoin keys and their owners, the Block chain parser breaks down each transaction in the Block chain and the key groups algorithm uses this data to produce groups of addresses. The schema for the database is listed below.

```
Username/Email & Key
{
	"_id":MongoDB Object Identifier
	"Username":String
	"Key":String
}
Transactions
{
	"_id":MongoDB Object Identifier
	"Inputs":[String]
	"Shadow Address":String
	"Outputs":[String]
	"Value In": Float
	"Shadow Value": Float
	"Output Value": Float
}

User & Key Groups
{
	"_id":MongoDB Object Identifier
	"Name":String
	"Addresses":[String]
	"Estimated Total": Float
}
```

The Username/Email & Key table contains a single entry for each user or company. It also stores a known key belonging to that user or company. The transaction table stores the most important information about each transaction in the Block chain. It stores the Block chain transaction ID as the object ID for the entry. It stores the Inputs to a transaction in an array of addresses, it stores the Shadow address as a string and the rest of the Outputs are stored in an array. It also stores the value for all of the Inputs, all of the Outputs and the Shadow address as a float. The third table is the User & Key Groups table. This table stores the Name of the user or company and an array of all addresses belonging to them. It also stores an estimate total of the balance belonging to this user. It is important to note that it will only be an estimate due to the database always being synced to a fixed time in the Block chain. This means that the users balance could have changed since the last time the transaction table was updated.

### 3.3.2. Crawler

One of the aims of my system is to remove some of the anonymity of the Bitcoin network by attaching either usernames or real names to groups of addresses. To do this it was essential that I created a database of users and their respective Bitcoin addresses. I decided that I would create a web crawler that would search the BitcoinTalk.org forums for any users who had a Bitcoin address on their public profile. I created a Python script using an external library called Beautiful Soup. This library is designed to make parsing HTML pages easier by turning the individual HTML tags into indexable objects. Using this I was able to search for the `<div>` tags corresponding to the username on the webpage. Once I had this `<div>` I was able to then lookup their user profile on BitcoinTalk.org and look to see if they have a Bitcoin address on show by looking for the `<div>` tag associated with it. To find the right `<div>` tag I wrote a for loop that would look at each `<div>` tag in the HTML document and check if it contained the CSS class corresponding to username and Bitcoin address. If an address was then found the address and username were then stored in the database in the User & Key table.

### 3.3.3. Block chain parser

I also needed to create a script that was capable of processing and storing transactions from the Bitcoin Block chain in a streamlined format. Since my application only needs the input and output addresses, that is all I store along with the transaction ID. This means that should I need to find out any additional information about the transaction that I have not got stored in my database I would be able to use the transaction ID to query the block and get the other data. Due to the fact that the Block chain is constantly growing I needed to create an algorithm that would be able to handle that. I decided that I would create a two part algorithm. The first part began at a fixed point in time and worked backwards through the Block chain. This script began at block number 281862 on the 22nd of January 2014. Its hash is "00000000000000000950dbc46b9c321f1c70afa49ea27ac9ddc6f927d63a04fde". The script took this hash and used the www.blockchain.Info API to request the details of it by using their raw block API method. This method returns all associated information about this block including its hash, its Merkel root, the hash of the previous block, a list of all transactions in the block, and the timestamp of the block along with a few checksums such as the nonce and size of the block.

This first script requests the block from the API, it then parses out the necessary information from the transaction list and stores it in the Transaction database. For each transaction the inputs, the input values, the outputs, the value of the outputs, the shadow address where the

change is returned to, the value of the shadow address, and the transaction id are stored in the transaction table. In the address information there will sometimes be a tag attached to the address. This is a name that has been attached to the address by the owner. I decided that I would add these addresses into the username & key table that my crawler had been populating. Once the block has been processed the previous block in the chain is then requested and processed by the script. This continues through the entire Block chain until it reaches the genesis block, the very first block mined by Satoshi.

The second part of the algorithm works from the current block back to the most recent block stored in the database. This is done by using the www.blockchain.Info API once again. There is a method on the API which returns the most recent block that they have confirmed and added to their database. I used this method to get the hash of the most recent block and then I worked backwards towards the starting address of the first part of the algorithm. Like the first part of the algorithm it stores the necessary information about the individual transactions in the transaction database and stores any addresses it finds with a tag in the Username & Key database.

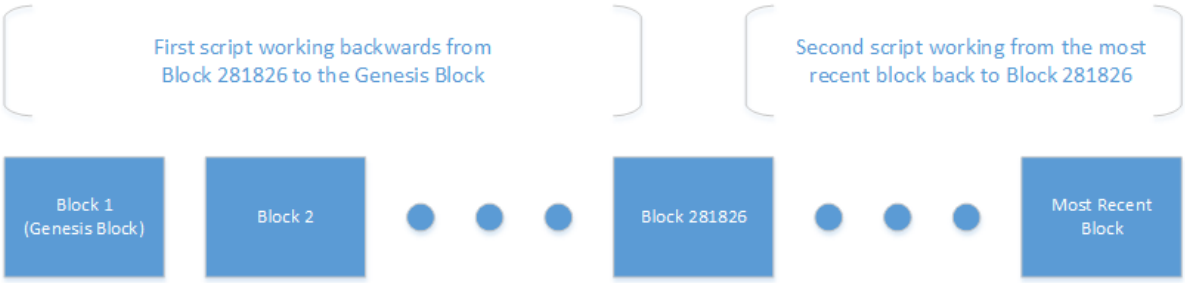


*Figure 24 Processing the Block Chain*

### 3.3.4. Group creation

Once the set of addresses and their respective owners had been created it was necessary to try and find any other addresses that belonged to the same user. To do this a script was created that would search the Block chain and determine if there are any addresses that can be grouped together and marked as belonging to one user. To do this it is necessary to utilize some properties of a Bitcoin transaction. The first property is that every transaction is sent from exactly one person but can be used to send coins to many people. This means that if Alice wants to pay Bob one Bitcoin she can do that in one transaction, similarly if Alice wants to send one Bitcoin to Bob and two Bitcoin to Charlie she can do that in one transaction. However if both Alice and Bob want to pay Charlie one Bitcoin each they would have to send it in separate transactions because their respective private keys would be stored on different nodes.

Without having both private keys it would not be possible to send a transaction from both Alice and Bob.

The second property of a Bitcoin transaction that is needed for this algorithm is that a transaction can have multiple input addresses. This means that if Alice needs to send Bob 5 Bitcoin but does not have 5 Bitcoin in one address. Instead Alice has 5 Bitcoin spread across three addresses. In this case Alice can use a multiple input transaction. This is a transaction where the three addresses that Alice has the Bitcoin dispersed in will be used as the inputs for the transaction to Bob. This is similar to how you can use multiple notes of varying value when buying something with fiat currency.

The third and final property of a transaction that I use in my solution is that when change needs to be returned to a user it is sent to the shadow address. This means that if Alice sends Bob 6 Bitcoin when she only owes him 5, the 1 Bitcoin that is left will be sent to a shadow address that Alice is the owner of. The user can sometimes specify their desired shadow address in their wallet application. If not then the wallet will generate one when it is sending the transaction and realizes that the user will need change.

Using these properties we are able to make some statements about the owner of various addresses in a transaction. Since a transaction can only be from one person we are able to say that the Input addresses all belong to the same user. This means that any multiple input transaction gives us several addresses belonging to one user. We can also conclude that the shadow address belongs to the same user as it is the mechanism used to deliver change in the Bitcoin network. It is not possible to say that the outputs are related due to the fact that a user can send Bitcoin to several different people in one transaction.

These properties were then used to produce an algorithm that creates a group of addresses belonging to a user. The algorithm begins by taking the known address and searching for any transaction that has it as an input or a shadow address. It is then added to the group list. All addresses in the transactions found that have not already been added to the group list are now added. Then for each address in the group list, a query to the database is made to find any transactions where it is present in the inputs or shadow address. Once again any addresses that are not in the group list are added to it. This loop continues until every address in the group list has been searched, including the addresses that are added as the loop is executing. Once all addresses have been searched then the group list is stored in the User & Key Groups table in the database.

```
groupList.add(StartAddress)
      for Address in groupList:
            results=db.search(inputs==(Address)||shadow==(Address))
            for potentialAddress in results:
                  groupList.add(potentialAddress)
      db.save(groupList)
```

*Group creation sudo code*

There were three different versions of this group creation script each operating on a slightly different data set. The first one began with the genesis block. It took the very first address in the very first block and tried to create a group of inputs and shadow addresses belonging to it. Once the first block was completed it moved on up through the Block chain. To keep track of addresses that have been searched a Python set was used. These sets are unordered collections and only allow unique items to be added. Python sets are implemented using a python dictionary and a hash function. The hash of the object being added to the set is stored in the dictionary as the key and the object is stored as the value. This means that look ups are very quick since you just need to hash the element you are looking for rather than searching element by element. It was for this reason combined with the fact that it only allows unique objects to be added, that a python set was chosen. When a new address is about to be searched it is first checked against the list of searched addresses. If it is present then it has already been searched and is skipped, otherwise the group creation algorithm runs.

The problem with this script is that due to the input data set being so large it would never have been able to finish it on the machine available. When timed it took on average 30 seconds to fully process one address. This is clearly far too slow for one machine to be able to process the entire Block chain. It was decided that the data set should be narrowed to create two smaller scripts. The first script only looked at the user & key table in the data base. Instead of looking at every address in the Block chain this script will only search for the addresses that I have a known user attached to. This database totalled 16,000 entries so it was decided that a second script would be used to look specifically at one user that was known to have many addresses. Mt. Gox the Bitcoin exchange that had closed down with 200,000 Bitcoin missing was chosen as the target for this script. This script worked in the same way as the others by beginning with one address known to belong to Mt. Gox and then searching for transactions where it is an input or shadow address.

It was important to also create a script to verify that these groups were correct. This script went through each group in the database and did a search for every address in the group. It would then create a set of the unique addresses using the search results and compare it to the original group. If the two matched then the group had been verified and was correct. If the two did not match then it was deemed that the set was invalid and it is marked in the database so that it could later be removed. When this script ran it did not find any invalid groupings so to test it an invalid group was added to the database to ensure it would flag the entry. An existing group was duplicated and an address that was known to not belong to the user involved and was not part of that transaction was added. Once the verifier script reached this entry it correctly identified it as an invalid grouping and marked it for removal.

### 3.3.5. Database schema modifications

There would only need to be two modifications to the database that I have made for the system to work. The first is to create a table that stores ratings on a user to user basis. This would need to store the user that is giving the rating, who they are giving the rating to and what the rating is. The second is to add an entry in the User and Key groups table. This entry would be used for storing ratings and would be a tuple of the user who gave the rating and their rating. That would make the table look as follows:

```
User & Key Groups{
       "_id":MongoDB Object Identifier
       "Name":String
       "Addresses":[String]
       "Estimated Total": Float
       "Ratings":[{“Name”:String, “Rating”:Integer}]
}
```

# 4. Experimentation

In this section I will discuss the closure of "Mt. Gox", the Bitcoin exchange, and I will attempt to analyse beyond the media speculation by utilising the tools I have created. By taking an address known to belong to Mt. Gox it will be possible to run it through the portion of the system that I implemented to identify linked addresses and get a summary of the transaction flow between them. I will go on to evaluate how the system performed during these tests and I will provide some predictions as to how the system would perform when scaled up to run on a cloud service like Amazon's EC2.

## 4.1. Transaction Malleability

In February of 2014 a bug was discovered in the Bitcoin protocol and source code produced by the Bitcoin foundation. [42] This code was used by dozens of exchanges and once discovered, a large number of them were forced to temporarily cease trading. One of the exchanges that was affected by this was Mt. Gox, the largest Bitcoin exchange, based in Tokyo Japan. The bug was known as "transaction malleability". This meant that when a transaction was signed by the sender, parts of the transaction could be changed without altering their signature. This meant that the transaction would receive a new transaction ID once it was added to a block and the signature would still appear to be valid. This means that any transactions that are relying on an output of this bogus transaction would be invalidated. It was not possible for the recipient or the amount fields of the transaction to be changed using this bug. It was possible however to change some fields meaning that chains of unconfirmed transactions could be broken by one transaction being modified. The modification of transactions in this manner caused a large number of unconfirmed transactions to build up while nodes struggled to correctly verify them. Many of the exchanges were affected by this bug however Mt. Gox is the only one to claim any losses linked to it.

Transaction malleability becomes an issue when a large chain of transactions are being created in a very short amount of time. In the case of Mt. Gox they used a system where every time a withdrawal was made the exchange sent coins from one of their wallets to the output address specified by the user. They used the same wallet address as their shadow address. This meant that the next time a user made a withdrawal, the previous transaction had to be referenced as an input to Mt. Gox's wallet. This means that if the transaction ID changed, the wallet would

be referencing an invalid transaction and would appear to be empty. Any transaction following this in the chain would also be rendered invalid and would be rejected by the Bitcoin network.

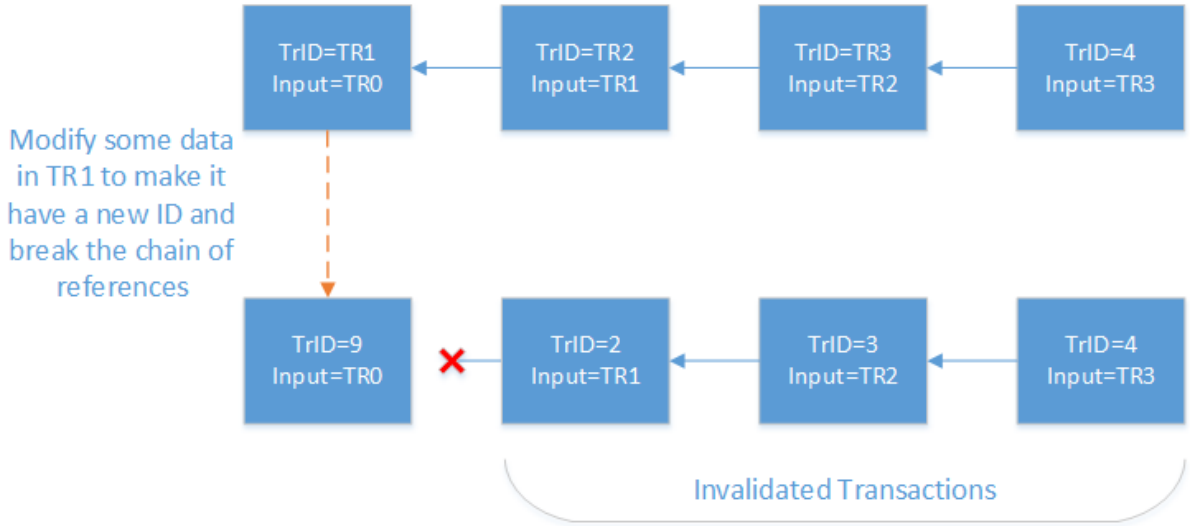


*Figure 25 An example of transaction malleability where a chain of unsigned transactions are broken due to a changed TrID*

A transaction malleability attack is slightly different. The attack involves tricking the original sender into thinking that the receiver never got the Bitcoin in the hope that they will send them again. For example if Alice send Bob 5 Bitcoin in transaction 10 and Alice see that it has an ID of 10. Before this block is signed Bob manages to change the ID to 101 and this is the final ID of the transaction once it is signed. Bob can then tell Alice that transaction 10 never happened and if Alice looks up the Block chain she will see that there is no transaction 10 only a transaction 101. If Alice falls for the trick she will send the coin again in a new transaction and Bob will have received double the amount of Alice's Bitcoin that he was supposed to.

Transaction malleability is what Mt. Gox claimed caused them to file for bankruptcy in February 2014, claiming that attackers had used transaction malleability attacks to drain their wallets leaving them with nothing. However there were some peculiarities surrounding these bankruptcy claims. Mt. Gox claimed that all of the coins they lost were due to the transaction malleability bug, however Christian Decker and Roger Wattenhofer, researchers from ETH Institute of technology in Zurich claim that any Bitcoin that was transferred out of Mt. Gox addresses happened after they halted transactions due to this bug. By analysing the Block chain they show that nearly all of the outgoing transactions happened after Mt. Gox blocked withdrawals. [43] They show a small number of transactions with similar attributes to a

transaction malleability attack before they ceased trading on the 8th of February, however these transactions only count for 1811.58 Bitcoins stolen from Mt. Gox. They then show that after Mt. Gox made a press release on the 10th of February blaming transaction Malleability attacks for their problems, the number of these transaction continued to increase to a point where close to 300,000 Bitcoins were removed from Mt. Gox wallets versus the 850,000 Bitcoin claimed to be missing.

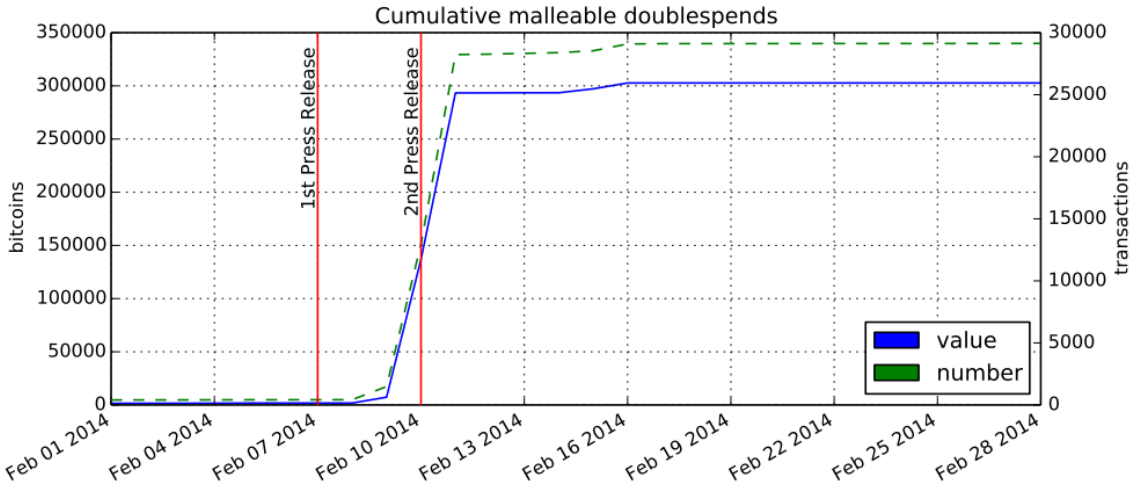


*Figure 26 Number of Malleability attacks during Mt. Gox's shutdown* [43]

## 4.2. Analysis of Mt. Gox

Due to the controversial nature of Mt. Gox and the disappearance of their coins I decided it would be interesting to use the tools I had created to examine some of their addresses in an effort to gain some insight into what really happened. I started by first finding an address that belonged to Mt. Gox. [44] One was found by my Block chain parsing tool when an address came in that had been tagged as Mt. Gox. I verified this on www.blockchain.info and Blockexplorer.com. From here I then used my group creation algorithm to create a group of addresses belonging to Mt. Gox. I was forced to stop this algorithm from completing due to time constraints. I had found a total of 213,699 addresses related to the address assumed to belong to Mt. Gox. Each lookup that the group creation algorithm used took on average 30 seconds per address. For this algorithm to complete all 213,699 addresses it would take 74.2 days to complete. By the time it reaches the end of those 74.2 days there would probably be countless more addresses than what was already in the database and searching these addresses could yield even more results. A calculation like this would need to be distributed across many nodes in a cloud infrastructure for it to be feasible.

Once I decided to stop this algorithm it was then necessary to process each address in an effort to discover how much Bitcoin was still held in the addresses linked to our original address. This was done by querying each address in the www.blockchain.info API. The results of these queries were then accumulated and a total value for the 213,699 addresses was created. The total number of Bitcoin found in these addresses is 117045.1649 which at the time of writing equates to roughly $52 Million. During their bankruptcy filings Mt. Gox found 200,000 coins of the missing 850,000. It was stated in a press release that these coins were moved to a new offline wallet. [45] If the wallet was a new one that they had not used before then my algorithms would not have created a link between their offline storage and the rest of the addresses they own. My analysis would suggest that the group of addresses that I found contain 117045.1649 Bitcoin. If these addresses do belong to Mt. Gox then there are two possible conclusions, first that the coins that were found were moved to an old wallet that had been used before by Mt. Gox or second that there are still some addresses belonging to Mt. Gox that do still contain Bitcoin.

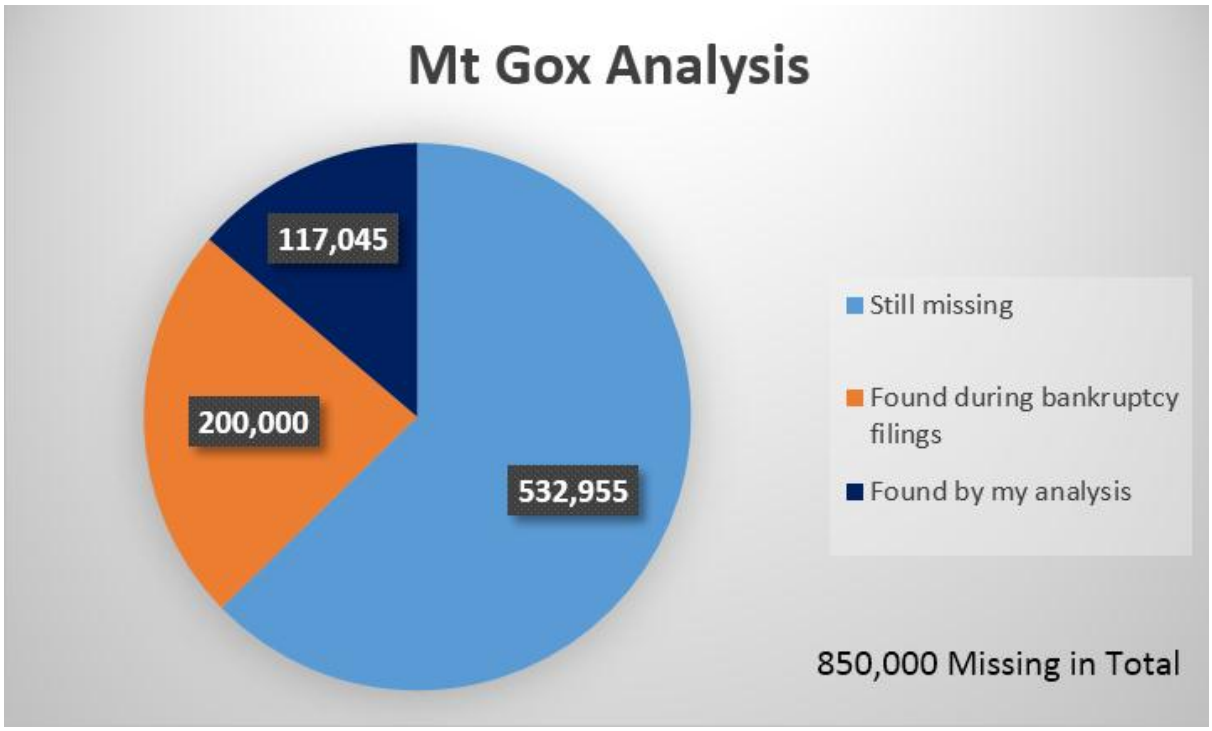


*Figure 27 Pie chart representing the missing Mt. Gox coins*

# 5. Conclusion

In this dissertation I discussed the technology behind Bitcoin and how it works at a low level. I discussed the history of Bitcoin and mining technology as it evolved to modern day ASICs. I made comparisons between Bitcoin and other crypto currencies and I gave a detailed account of some of the media controversies surrounding Bitcoin. I then outlined the advantages of some of the technologies I used in both my design and implementation. I set out the design a system to provide more detailed information about the Bitcoin Blockchain. The system I designed would be able to provide more detailed information to government bodies and revenue agencies. By providing a detailed transaction history for a user and an estimated total balance it would help financial regulators with enforcing compliance. This would allow financial regulators to attempt to enforce regulation and taxation on Bitcoin by using this system. Using only one address belonging to a user the system can provide a group of addresses that are linked to that address. From there the user's entire transaction history, their reputation score and estimated value would be visible to the regulator in the designed system. It would also be possible to see any malicious behaviour from any of the users addresses such as any double spend attacks or transaction malleability attempts they have made.

Not only would this system be useful for financial regulators but it would also be useful for users of the Bitcoin community. The system would allow users to see the same information as stated above and it would also have the added functionality of facilitating ratings using the design I outlined in the beginning of my design chapter. The user can claim an address as their own and say that they want to rate a given transaction. The system would be capable of using the underlying public and private keys associated with a user's Bitcoin address for encrypting and validating the user's rating. This system would also only allow users who have interacted with one another to give each other feedback and this in turn would help prevent fake reviews. This is because of the miner's fee included in Bitcoin meaning that users would have to pay to rate themselves.

This system would rely on the scripts and algorithms that I discussed in the second part of my design chapter. The web crawler script which would be used to gather a list of usernames and Bitcoin addresses belonging to them, the Block chain parser script for getting the most recent blocks from the Block chain to keep the system up to date and the group creation algorithm to create the groups of addresses that each user owns. Together these three scripts can create the

data that would provide the basis for the system that I outlined in the System Design section of my design chapter.

## 5.1. The future of Bitcoin

It is very difficult to speculate whether Bitcoin as it is currently used today will still be around in 20 years' time. There are some interesting advances that could lead to many things being built on top of the Bitcoin network. I think that it is more likely that these advances are what will be around for longer and that Bitcoin as a currency may lose some of its attention once it has reached its maximum number of Bitcoins in circulation. Some of the interesting advances are building applications that utilise the Bitcoin platform in some way. One example of this is an advancement on the Bitcoin infrastructure similar to the authentication method I described in my design chapter. Using an existing network of Bitcoin public keys and addresses to authenticate a user without the need for passwords or usernames. It has been said that Bitcoin as a currency is only the first application of the network. By creating this vast peer to peer proof of work network users will find a way to use it for many other things. I think that the most important part of Bitcoin is not its use as an alternate currency. The most important part is its distributed peer to peer network capable of signing and agreeing upon a chain of decisions. Since it is open source it will allow developers to use the computation power of this network for many different applications beyond using Bitcoin as a currency.

## 5.2. Other possible advances

The only issue that I experienced with these programs is time. They take a very long amount of time to run due to the large amount of data that must be searched through. Some optimisations could be made on this front. By using faster data structures and parallelising the algorithms it would be possible to increase the data throughput of each of the algorithms. By using hash tables to store the data it would make lookups much faster allowing the algorithms to run even faster. It would also be advantageous to optimise the programs for multicore platforms so as to take advantage of running the application in parallel. In my design I discuss using a cloud platform for the system I outlined. By using a cloud platform the system could elastically expand as more users tried to use the system, but more importantly if the system was properly parallelised many cheaper nodes could be used to do the work of a few expensive nodes.

# References


[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” pp. 1-9, 2009.

[2] F. Reid and M. Harrigan, “An Analysis of Anonymity in the Bitcoin System,” *Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on Social Computing,* vol. 3, pp. 1318-1326, 2011.

[3] D. Knuth, The art of computer programming: Volume 3, Reading, Massachusetts: Addison-Wesley, 1973.

[4] T. S. Phillip Rogaway, “Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance,” *11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004.,* vol. 3017, pp. 371-388, 2004.

[5] R. Merkle, “A certified digital signature,” in *Crypto*, Berlin, 1989.

[6] NIST, “Descriptions of SHA-256, SHA-384, and SHA-512,” NIST.gov, 2001.

[7] H. Dobbertin, “The Status of MD5 After a Recent Attack,” RSA Laboratories newsletter CryptoBytes, Cambridge, MA, 1996.

[8] B. Schneier, “Cryptanalysis of SHA-1,” 18 February 2005. [Online]. Available: https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html. [Accessed 14 May 2014].

[9] H. Gilbert and H. Handschuh, “Security Analysis of SHA-256 and sisters,” *10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003. Revised Papers,* vol. 3006, pp. 175-193, 2003.

[10] Bitcoin Foundation, “Transaction Fees,” 19 December 2010. [Online]. Available: https://en.bitcoin.it/wiki/Transaction_fees. [Accessed 28 March 2014].

[11] Bitcoin Foundation, “Block Hashing Algorithm,” 19 December 2010. [Online]. Available: https://en.bitcoin.it/wiki/Block_hashing_algorithm. [Accessed 14 April 2014].

[12] M. B. Taylor, “Bitcoin and The Age of Bespoke Silicon,” *International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pp. 1-10, 2013.

[13] B. F. “Bitcoin Wiki Mining Hardware Outputs,” 24 December 2010. [Online]. Available: https://en.bitcoin.it/wiki/Mining_hardware_comparison#Graphics_cards. [Accessed 30 4 2014].

[14] M. J. S. Smith, Application-Specific Integrated Circuits, New York: Addison-Wesley Professional, 1997.

[15] H. Esmaeilzadehy, E. Blemz, R. S. Amantx, K. Sankaralingamz and D. Burger, “Dark silicon and the end of multicore scaling,” *International Symposium on Computer Architecture*, vol. 38, pp. 365 - 376, 2011.

[16] “Butterfly Labs Bitcoin ASIC manufacturers,” [Online]. Available: http://www.butterflylabs.com/. [Accessed 18 April 2014].

[17] Bitcoin Foundation, “Bitcoin Addresses,” 22 November 2011. [Online]. Available: https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses. [Accessed 28 March 2014].

[18] X. Wang and H. Yu, “How to Break MD5 and Other Hash Functions,” *EUROCRYPT*, vol. 3494, pp. 19-35, 2005.

[19] CERT, “CERT Software Engineering Institute,” CERT, 31 December 2008. [Online]. Available: http://www.kb.cert.org/vuls/id/836068. [Accessed 24 April 2014].

[20] I. Eyal and E. G. Sirer, “Majority is not Enough: Bitcoin Mining is Vulnerable,” *Computing Research Repository - Cornell University*, vol. 1311.0243, pp. 1-17, 2013.

[21] J. Stevenson, “Getting started with Litecoin,” Self, 2013, p. 6.

[22] C. Percival, “Stronger key derivation via sequential memory-hard functions,” *The BSD Conference (BSDCan)*, pp. 1-19, 2009.

[23] A. Barsuhn, S. Lucks and C. Forler, “Cache Timing Attack on Scrypt,” in *19th Crypto Day*, Stuttgart, Germany, 2012.

[24] P. F. “Peercoin,” 2014. [Online]. Available: http://www.peercoin.net/. [Accessed 25 4 2014].

[25] “Bitcoin Talk,” 1 March 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=3984. [Accessed 22 April 2014].

[26] T. T. project, “About TOR,” [Online]. Available: https://www.torproject.org/about/overview.html.en. [Accessed 26 04 2014].

[27] K. Hill, “Forbes,” 16 January 2014. [Online]. Available: http://www.forbes.com/sites/kashmirhill/2014/01/16/the-feds-are-ready-to-sell-the-silk-road-bitcoin-kind-of/. [Accessed 22 April 2014].

[28] E. F. F. “TOR,” 2013. [Online]. Available: https://ssd.eff.org/tech/tor. [Accessed 26 4 2014].

[29] R. Wile, “Business Insider,” 27 November 2013. [Online]. Available: http://www.businessinsider.com/bitcoin-1000-2013-11. [Accessed 23 April 2014].

[30] D. J.-P. Rodrigue, “The Geography of Transport Systems - Stages in a Bubble,” 2009. [Online]. Available: http://people.hofstra.edu/geotrans/eng/ch7en/conc7en/stages_in_a_bubble.html. [Accessed 26 4 2014].

[31] “Bitcoin Talk,” 22 October 2013. [Online]. Available: https://bitcointalk.org/index.php?topic=316152.0. [Accessed 22 April 2014].

[32] J. WEISENTHAL, “Business Insider,” 7 February 2014. [Online]. Available: http://www.businessinsider.com/mtgox-halts-withdrawals-2014-2. [Accessed 22 April 2014].

[33] A. Hern, “The Guardian,” 27 February 2014. [Online]. Available: http://www.theguardian.com/technology/2014/feb/27/how-does-a-bug-in-bitcoin-lead-to-mtgoxs-collapse. [Accessed 28 April 2014].

[34] “BBC.co.uk,” March 2014. [Online]. Available: http://www.bbc.com/news/technology-26677291. [Accessed 22 April 2014].

[35] J. Yarow, “Business Insider,” 6 February 2014. [Online]. Available: http://www.businessinsider.com/why-apple-is-anti-bitcoin-apps-right-now-2014-2. [Accessed 28 April 2014].

[36] R. Banagale, “Gliph Blog,” Gliph, 9 December 2013. [Online]. Available: https://blog.gli.ph/2013/12/09/the-state-of-bitcoin-mobile-applications-in-the-app-store-and-google-play/. [Accessed 28 April 2014].

[37] R. A. Green, “Forbes,” 3 December 2013. [Online]. Available: http://www.forbes.com/sites/greatspeculations/2013/12/03/the-tricky-business-of-taxing-bitcoin/. [Accessed 28 April 2014].

[38] Google, “Google app engine documentation,” Google, 15 April 2014. [Online]. Available: https://developers.google.com/appengine/docs/python/. [Accessed 3 5 2014].

[39] A. Whitby, A. Jøsang and J. Indulska, “Filtering Out Unfair Ratings in Bayesian Reputation Systems,” in *The Third International Joint Conference on Autonomous Agenst Systems*, New York, 2004.

[40] Internal Revenue Service, “IRS Bitcoin announcement,” 18 March 2014. [Online]. Available: http://www.irs.gov/pub/irs-drop/n-14-21.pdf. [Accessed 5 May 2014].

[41] Amazon, “Dynamo DB,” Amazon.com, 2014. [Online]. Available: http://aws.amazon.com/dynamodb/. [Accessed 21 April 2014].

[42] B. F. “Transaction Malleability,” 11 February 2014. [Online]. Available: https://en.bitcoin.it/wiki/Transaction_Malleability. [Accessed 5 May 2014].

[43] D. Christian and W. Roger, “Computing Research Repository - Cornell University,” 26 March 2014. [Online]. Available: http://arxiv.org/pdf/1403.6676v1.pdf. [Accessed 5 5 2014].

[44] Blockchain.info, “Mt. Gox Address,” [Online]. Available: https://blockchain.info/address/1LNWw6yCxkUmkhArb2Nf2MPw6vG7u5WG7q. [Accessed 19 May 2014].

[45] Mt. Gox, “Mt. Gox press release,” 20 March 2014. [Online]. Available: https://www.mtgox.com/img/pdf/20140320-btc-announce.pdf. [Accessed 19 May 2014].

[46] J. B. M. Y. J. Xu, “A generalization of the beta distribution with applications,” *Journal of Econometrics*, vol. 66, no. 1-2, pp. 133-152, 1995.

[47] K. DRAWBAUGH and P. TEMPLE-WEST, “Reuters.com,” 25 March 2014. [Online]. Available: http://www.reuters.com/article/2014/03/25/us-bitcoin-irs-idUSBREA2O1LR20140325. [Accessed 4 May 2014].

[48] T. Boice, “Crypto Coin News,” 23 April 2014. [Online]. Available: http://www.cryptocoinsnews.com/news/blockchain-voting-used-by-danish-political-party/2014/04/23. [Accessed 6 May 2014].

[49] K. Shiriff, “List of Blockchain messages,” 16 February 2014. [Online]. Available: http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html. [Accessed 6 May 2014].

[50] K. Torpey, “BitID,” 5 5 2014. [Online]. Available: http://www.cryptocoinsnews.com/news/forget-logging-facebook-bitid-lets-sign-bitcoin-address/2014/05/05. [Accessed 6 May 2014].