Bitcoin:
Digital Currency for the 21st Century

Nick Kamper (kampernj@rose-hulman.edu)
Mike McDonald (mcdonamp@rose-hulman.edu)
Ryan Oliver (oliverr@rose-hulman.edu)
Stephen Shinn (shinnsm@rose-hulman.edu)

May 12, 2013

Contents

1 Bitcoin Background

The Bitcoin paper was published in 2008 by Satoshi Nakamoto, which is a pseudonym for an individual or group of developers who created an open source peer to peer digital currency, and the first open source client was released shortly after [**?**]. Bitcoins started off as relatively useless, but have risen to a current value of around $110 USD, and have begun to be accepted by many online retailers, though no "mainstream" companies have caught on yet (presumably due to large fluctuations in the currency and gaps in the governmental regulation of virtual currencies).

Bitcoins were first created through the process of "mining" them, but they can be transferred as well. Mining is what allows the system to remain secure and deal with issues of double-spending while processing all transactions that are submitted. The basis of mining is founded on the Hashcash proof-of-work (POW), with a few innovative extensions to make the system more secure and controllable. However, in order to maintain its security, there must be more honest groups working on POW problems than malicious groups. In order to encourage people to mine, Bitcoin rewards a dynamic number of Bitcoins plus the transaction fees to each person or group who finds a valid POW.

POWs are generally in place to deter denial of service–spam–attacks by requiring the service requester to do some work for each request. It is designed so that spammers are discouraged, but normal users don't feel the effect of the processing time required. The work must be reasonably difficult to do, and Hashcash actually requires brute force, but also must be easy to check so that the server doesn't need to spend much processing time confirming the work completed was valid [1].

1.1 Hashcash and Bitcoin

An example of where Hashcash could be used is through email. The sender prepares a header to their email and adds an initial random number called a nonce. They then compute the SHA-1 hash of their header and look at the first 20 bits of the hash. If they are all zeros, the header is acceptable. If not, the sender increments the nonce by 1 and tries again. Since the output of SHA-1 can't be predicted, this forces the sender to brute force finding an acceptable header. Since there are 1 in $2\hat{2}0$ headers that start with twenty 0's, the sender will have to calculate $2\hat{1}9$ hashes on average to find a valid header. This takes approximately 1 second to currently compute, keeping lag time minimal while decreasing the likelihood that spammer will send large amount of mail. On the other hand, the recipient doesn't have to complete very much work at all. Once they receive the header of the email, they calculate the SHA-1 hash and check if the first twenty bits are 0's. They then check to see if the date is within two days allowing leeway for clock skews and routing time and check if the e-mail

address in the header is valid. If everything is valid, as long as the email wasn't already in the database, it is then added. The computation time computing SHA-1 once is relatively small and so this system fulfills the general purpose of the POW concept [1].

Bitcoin uses a very similar POW. However, improvements have been made for increased security as well as inflationary control of a Bitcoin's worth. SHA-1 has been shown to be insecure and it is expected within the next five years for organizations to have enough computation power to crack SHA-1 by brute force [2]. For this reason, Bitcoin replaced SHA-1 with SHA-256. It also runs the hash twice for added security measure. Inflationary control is also a large worry for this decentralized digital currency. As computation techniques and hardware improve, more and more computational power can be put into mining and would be able to find solutions to the POW much faster than desired. This would lead to Bitcoins being rewarded faster than is healthy for a monetary supply. To overcome this, Bitcoin has a dynamically adjustable difficulty which changes the number of leading zeros needed for a valid POW. It takes a moving-average of the number of valid POWs found per hour and keeps the rate relatively constant no matter how little or how much computation power is being inputted. This allows for a controllable number of Bitcoins to be rewarded over time, leading to better inflationary control of a virtually scarce source, which provides value to the currency [3].

2 Transferring Bitcoins

The most common way Bitcoins are acquired is through transferring them from one individual to another, through an electronic Bitcoin "wallet". One Bitcoin is defined by a chain of Elliptic Curve Digital Signal Algorithm (ECDSA) digital signatures[1]. Every Bitcoin owner has both a public and private key. To transfer a coin from one owner to the next, the current owner uses their private key to digitally-sign a double SHA-256 hash of (1) the previous transaction and (2) the public key of the next owner[2]. This value is then added to the end of the coin, and the process is repeated for subsequent transactions. Figure 1 illustrates the transaction process[3].

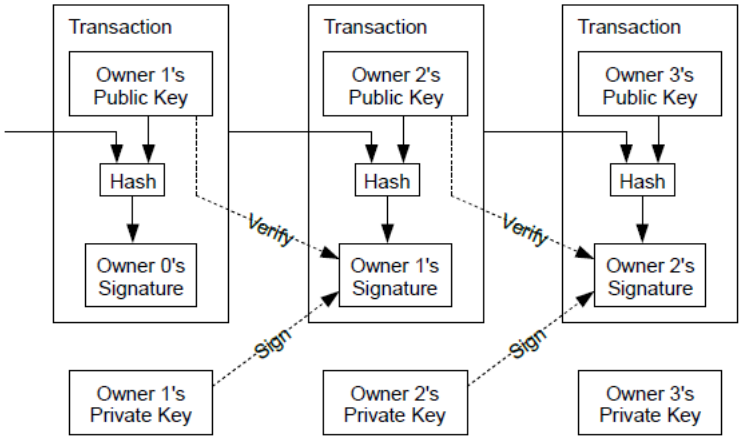

Figure 1: Bitcoin Transaction Process

To prevent having to make a transaction for every cent in a transfer, transactions contain multiple inputs and outputs to allow value to be split and combined. There is normally either a single input from a larger previous transaction, or multiple inputs combining smaller amounts. There are at most two outputs: one for the payment and one for returning change, if any, back to the sender[3].

2.1 What is a Bitcoin "wallet"

In practice, a Bitcoin "wallet" is what allows users to send and receive Bitcoins. A wallet provides ownership of one or more Bitcoin addresses, which can be created as needed. Addresses are stored in a wallet file with links to cryptographic passwords (the ECDSA private keys) which enable bitcoin transfer[4]. There are three types of wallets available: software, mobile, and web[4]. Figure 2 shows an example of a software wallet called Electrum, with a simple user interface for sending and receiving Bitcoins.

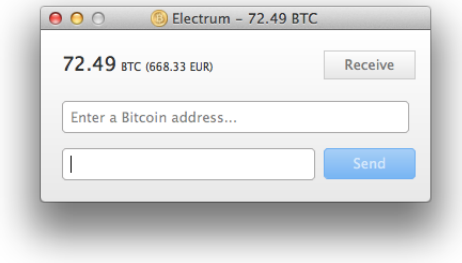

Figure 2: Electrum Bitcoin Wallet

Wallets also keep track of a user's transaction history and balance. Figure 3 shows a sample of transaction history in the Electrum wallet.

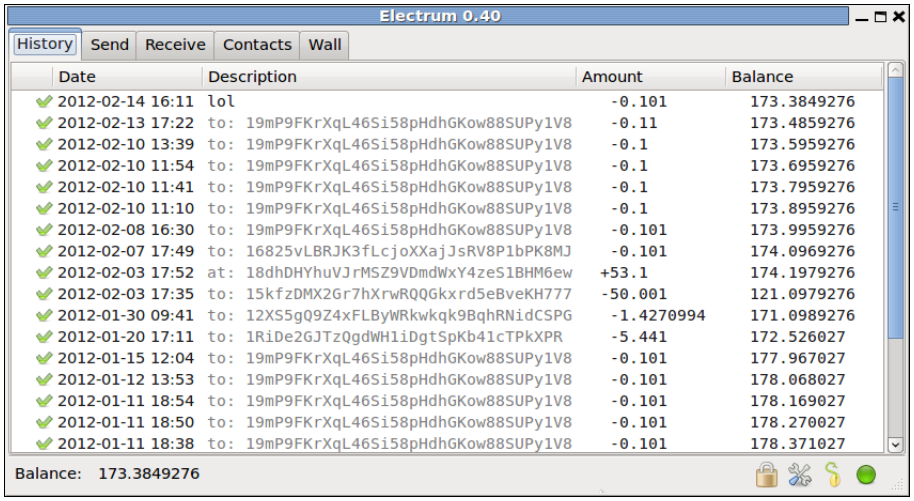

Figure 3: Electrum Transaction Log

2.1.1 How are addresses calculated?

A Bitcoin address is a 160-bit hash of the ECDSA public key used in the transaction process. The address is generated through the process described below, which includes an example address generation along the way[5]:

1. Start with the ECDSA public key.

   ```
   0450863AD64A87AE8A2FE83C1AF1A8403CB53F53E486D8511DAD8A04887E5B235
   22CD470243453A299FA9E77237716103ABC11A1DF38855ED6F2EE187E9C582BA6
   ```

2. Perform SHA-256 hashing on the public key.

   ```
   600FFE422B4E00731A59557A5CCA46CC183944191006324A447BDB2D98D4B408
   ```

3. Perform RIPEMD-160 hashing on the result of SHA-256.

   ```
   010966776006953D5567439E5E39F86A0D273BEE
   ```

4. Add version byte in front of RIPEMD-160 hash (0x00 for Main Network).

   ```
   00010966776006953D5567439E5E39F86A0D273BEE
   ```

5. Perform SHA-256 hash on the extended RIPEMD-160 result.

   ```
   445C7A8007A93D8733188288BB320A8FE2DEBD2AE1B47F0F50BC10BAE845C094
   ```

6. Perform SHA-256 hash on the result of the previous SHA-256 hash.

   ```
   D61967F63C7DD183914A4AE452C9F6AD5D462CE3D277798075B107615C1A8A30
   ```

7. Take the first 4 bytes of the second SHA-256 hash. This is the address checksum.

   ```
   D61967F6
   ```

8. Add the 4 checksum bytes from point 7 at the end of extended RIPEMD-160 hash from point 4. This is the 25-byte binary Bitcoin address.

   ```
   00010966776006953D5567439E5E39F86A0D273BEED61967F6
   ```

9. Convert the result from a byte string into a base58 string using Base58Check encoding. This is the most commonly used address format.

   ```
   16UwLL9Risc3QfPqBUvKofHmBQ7wMtjvM
   ```

2.2 Verification and Double Spending Prevention

Without a central authority, Bitcoin must protect users from double-spending by verifying the authenticity of both the coin and the owner of the coin. One way it does this is from implementing a public transaction log displaying all past transactions. It holds cryptographically enforced append-only properties which means it doesn't allow nodes to change the past and double-spend Bitcoins or gain others Bitcoins maliciously. This is enforced over the Bitcoin person-to-person network (p2p) where there is a distributed management approach to the security of the log. Due to this model, the Byzantine General problem must be overcome. This basically calls for ways to prevent a few malicious users (nodes) from harming the network and/or the log and causing havoc such as by rewriting the public log, stealing Bitcoins, etc. The reward system in place is expected to encourage enough benevolent nodes to be computing POWs to overcome the POWs created by the malicious nodes. In order to see this more clearly, the public log system must be examined in further detail[6].

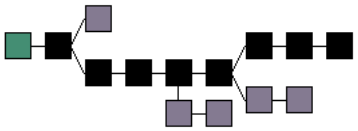

Figure 4: Blocks of the public log chain

The public log system is a linear chain of blocks which hold, among other things, time-stamped transactions. This chain is continuously being built on through the hash-based POWs. When a POW is considered valid, the block is added to the chain and is considered written in stone. It, along with past records, can't be changed without redoing the POW and all POWs following the block that is being changed. The likelihood of this happening and the system accepting this changed chain of POWs as valid is small. This is because the longest chain of blocks is considered the correct record of all past events and is from the largest pool of computing power. While people may try to rewrite blocks in the past, as long as there are more benevolent nodes that use their computing power to extend the true chain, it makes it more and more difficult for malevolent nodes to catch up since the more numerous good nodes will generate longer chains more quickly than the attackers. This system requires minimal structure since all transactions are broadcast and nodes can leave or rejoin the network at will. Note that even if a transaction isn't broadcast to every node, it will eventually be used in a block. Also, as a node joins, it accepts that longest chain as the record of events that happened since they left and then they continue building onto it. Similarly, if a node misses a block, it notices that its reference to the previous block doesn't match and so requests the blocks that are missing[6].

2.3 Attacks on Bitcoin

Even if a malevolent attack was able to start constructing a chain faster than the benevolent nodes, the system still limits what damage could be caused. Nodes still must accept the new blocks as valid and so if there are transactions such as spending Bitcoins that aren't actually there or spending someone else's Bitcoins, the nodes will reject the new block. The main reason someone would attack the system would be to change a previous transaction made by the attacker and then re-spend the Bitcoins. While this would lead to the problems of double-spending that Bitcoin tries to eliminate, the effects would still be rather limited. However, it is still very unlikely than an attack will be able to complete POWs faster than benevolent nodes. This race can be expressed as Binomial Random Walk where a benevolent node successfully creating a POW increases the lead by 1 and the failure of a malevolent node creating a valid POW reduces the lead by 1. The probability that the malevolent node catches up to the benevolent nodes can be calculated from the Gambler's Ruin problem[6]:

p = probability an honest node finds the next block
q = probability an attacker finds the next block
q_z = probability an attacker will ever catch up from z blocks behind

$$q_z = \begin{cases} 1 & : p \leq q \\ \left(\frac{q}{p}\right)^z & : p > q \end{cases}$$

Since p is generally larger than q, unless the attacker is lucky in the beginning, the chance that the attacker catches up decreases exponentially. If, for example, an attacker started working on a new chain of blocks from one that was just completed, to calculate how many blocks would need to be added to the chain to so that the attacker has less than a 0.1% chance of catching up, a Poisson distribution can be used. The expected value is as follows and is used to calculate the probability of the attacker catching up[6]:

$$\lambda = z\left(\frac{q}{p}\right)$$

$$P(z) = \sum_{k=0}^{\infty} \frac{\lambda^k e^{-\lambda}}{k!} q_z$$

$$P(z) = 1 - \sum_{k=0}^{z} \frac{\lambda^k e^{-\lambda}}{k!} \left(1 - \left(\frac{q}{p}\right)^{z-k}\right)$$

For different values of z and q, where p=1-q, are as follows. Notice how the probability drops off exponentially as expected[6]:

	q = 0.1	q = 0.3
z	**P(z)**	**P(z)**
0	100%	100%
1	20.46%	17.74%
2	5.10%	4.17%
3	1.32%	1.01%
4	0.35%	0.25%
5	0.09%	0.06%
6	0.02%	0.01%
7	0.01%	0%
8	0%	0%

Table 1: Expected value table for P(z) for different values of q

Solving for the number of blocks that must be found valid before the attacker no longer needs to be worried about provides the following data [3]:

q	**z**
0.1	5
0.15	8
0.2	11
0.25	15
0.3	24
0.35	41
0.4	89
0.45	340

Table 2: Table of q and z values such that P(z) $< 0.1\%$

Further calculations and code can be found in the official Bitcoin paper found at source[6].

Another way in which an attack might try to change the system is to come up with a chain in advance that they would try to append on the current public chain. However, as in the more detailed public log diagram below, each block has a header, which is what is hashed to find valid POWs, that includes hashes of the transactions. Since the attacker can only change their own transactions, they won't know what the header file looks like until the public key is sent to them for the transaction. They also wouldn't be able to use any transactions in their blocks that have already been processed because they would then be considered invalid. At this point, the hash of the header will change and so any chain created in advance will be invalid. Even if they were trying to append blocks to the end of the chain that weren't changes from previous blocks, since the first block's header includes the previous blocks hash, the POWs would need to be recalculated since the header of the first block in the fake' chain would change as each valid block is appended to the actual chain[6].

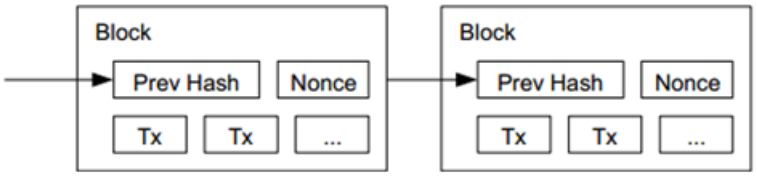

Figure 5: Public log system chain made of transaction blocks

To see how this works exactly, a more accurate description of a block in the public log is needed. A block is made up of the following[6]:

Magic Number	0xD9B4BEF9	4 bytes
Block Size	-	4 bytes
Block Header	See below	80 bytes
Transaction Counter	Number of Transactions	1-9 bytes
Transactions	Transactions and Hashes	As needed

Table 3: Bitcoin Block Contents

Version	Updates when software changes	4 bytes
Hash of previous header	Updates when new block received	32 bytes
Hash of Merkle Root	Updates when transaction accepted	32 bytes
Time	Updates every few seconds	4 bytes
Difficulty Target	Updates when difficulty changes	4 bytes
Nonce	Updates when a hash is tried (starts at zero)	4 bytes

Table 4: Bitcoin Block Header Contents

Additionally, we can zoom in on the transactions and uncover the Root node of the Merkle Tree, shown below.

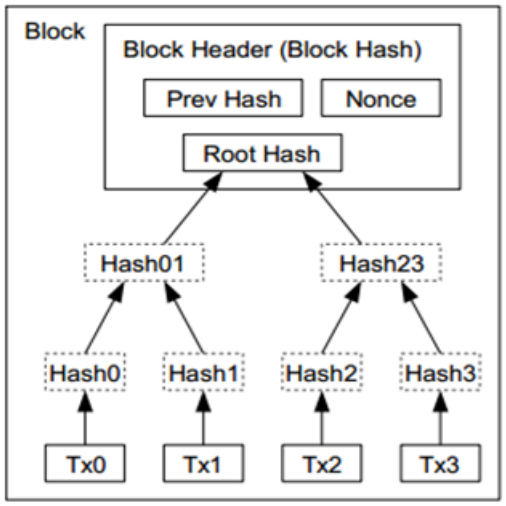

Figure 6: Transactions hashed in a Merkle tree

The Merkle Tree Root is in place so that there is a record of all transactions in the block header without actually needing the hash of every transaction in the header. This tree is created by using double SHA-256 and then starting at the bottom row of the tree and building up. The first row is formed by ordered hashes of all transactions in the block. The first hash is known as the generation hash and is from a transaction to type that starts a new Bitcoin and is addressed to the person solving the POW's. The row above consists of half of the number of hashes where each entry is the double SHA-256 hash of the concatenated 64-byte results of the row below. If a non-root row has an odd number of elements, the final hash is duplicated so that each row is even. This is continued until the final row has a single double SHA-256 hash, which is the Merkle root, and stores it in the header. The root lets the header to be much smaller and easier to hash and solve POWs for[6].

The nonce in the header is the number used to increment the header when trying to find a POW. Since it is so small, it often will overflow. In this case, it will increment the extraNonce variable of the generation transaction and will then restart incrementing the nonce. This allows for the continuous chance of the header until one that fulfills the difficulty requirements is found. One of the reasons this evaluation works well is because each person solving for the POW has a unique merkle root due to the generation transaction. This almost guarantees that each person is producing different hashes and allows for a valid POW to be found relatively quickly as many nodes search for it[6].

In order to verify payments, a user acquires a copy of the block headers of the longest proof-of-work chain by querying network nodes. The user then obtains the Merkle branch linking the transaction to the block it is timestamped in. By linking the transaction to a place in the chain, the user is able to confirm that a network node has accepted it–verifying the transaction.

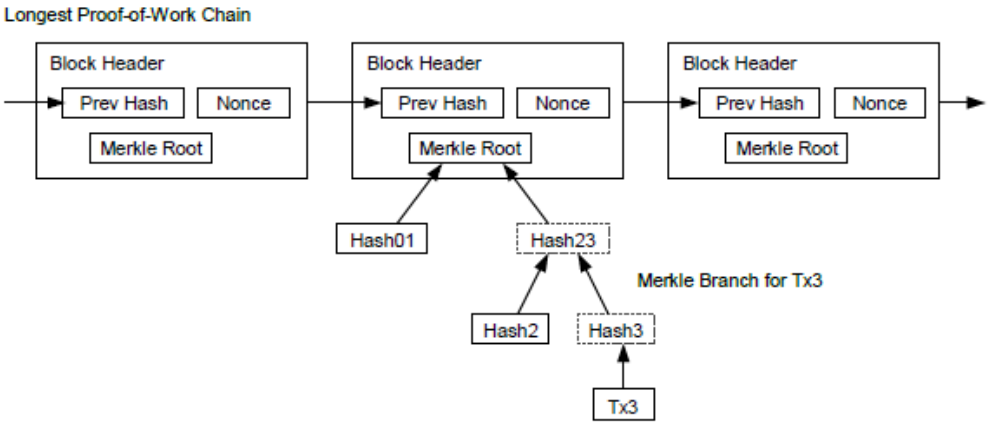

Figure 7: Verification of a transaction [3]

3 Mining Bitcoins

In addition to being bought and sold on online currency exchanges, Bitcoins can be mined by anyone with access to a computer and the ability to install software on it. The analogy to mining is essentially the following: just as men and women toil to extract precious resources from the ground, expending time and energy, Bitcoin mining programs require time and energy to solve POWs, thus creating value for the resources they have created. And, just like real mining, the tools, as well as the procedure used, have a large impact on the return on investment.

3.1 Bitcoin Mining Procedure

The mining process is directly related to the header, especially the nonce and Merkle root.

The general process is as follows[7]:

1. New transactions are broadcast to all nodes
2. Each node collects new transactions into a block
3. Each node works to find a POW for its block
4. Bitcoins are successfully mined by the receiving node which found the POW
5. Nodes accept the block if all transactions are valid, no double-spending occurs, and the POW checks out
6. Nodes begin to create the next block in the chain, using the hash of the accepted block as the 'previous hash' in the header. This shows their acceptable of the block as valid.
7. Repeat

More specifically, during step three, the miners continuously scan for a header that, when hashed with SHA-256, produces a hash that begins with a certain number of zeros based on the currently difficulty. The difficulty changes dynamically in order to keep block production to around six blocks per hour. If a header file doesn't work, the nonce is then incremented, changing the header and causing a different hash to be produced. This continues until a block is validated. If two blocks happen to be validated at the same time, some nodes will be sent one block and the rest will be sent the other. The first block that is extended creates the longest chain and is from then-on used as part of the public chain[6].

Once a block is validated, a reward of 50 Bitcoins, plus the transaction fees from the users whose transactions were included in the block, are paid. The reward, however, is a dynamic number and halves every 210,000 blocks that are

produced, which by estimates is about every four years. The current payout is at 25 Bitcoins. Eventually this will limit out to 21 million Bitcoins in existence. However, based on fractional reserve banking, the money supply could be much larger. This rate of mining and limitations on the supply were based on the approximate rate at which commodities like gold are mined and are hoped to mimic the relative stability of gold in the long run. Similarly to gold, the chance of someone mining a Bitcoin is relatively small. On one's own, it is possible to go months with mining a single Bitcoin. It is also important to note that mining is an energy intensive computation, so that even if a Bitcoin is mined, it is not always done at a profit[6].

3.2 Increasing Bitcoin earnings

3.2.1 Pooling Resources

In order to improve earnings, leaving less up to chance, many people have begun to bundle their computation power and pool them in central servers. Servers working alone take a relatively long time to have their block confirmed and to receive payment, both probability-wise as well as overall, since their small computation power doesn't provide a large percentage of the overall computation power. Pooling resources allow for all participating servers to receive a proportion of the Bitcoins earned every time the pooled server solves a POW and it gets validated. While the reward is generally proportional to the amount of time put in and percent of computation power provided, this process provides a steadier income, although much smaller than a single reward earned working solo. This pooling of resources has started concerning some researchers though.

For example, the collaboration called Deepbit currently provides approximately 40% of all computational power. If this network turned malicious, Harvard researchers estimated that it would take about 1 year, after doubling its market share, to revise the entire public chain, taking into account Moore's law. While it is improbable that such a network would turn malicious, research is currently taking place to limit such possibilities in the future[7].

3.2.2 Hardware

Another way to improve possible earnings is to purchase better hardware. Originally, the Bitcoin client used only the CPU for computations. However, since the CPUs main purpose is for decision making and not repetitive processes, GPUs soon took over. GPUs could process 3200 32-bit instructions per clock cycle versus the 4 or 8 32-bit instructions per clock cycles common to CPUs. This is because GPUs have a large number of ALUs and are designed to do repetitive tasks like video processing, and so finding POWs meshed well. With a single card, current GPUs like the model 5830 compute 302 million hashes

per second and only cost $92[8]. People have also tried to implement the computations on FPGAs since they generally consume small amounts of power and have relatively high computation abilities. Butterflylabs produces a Mini Rig, one of the highest rated FPGAs, that produces 25.2 billion hashes per second. However, it does cost $15,295 and consumes a large amount of power, so if comparing the number of millions of hashes per second per dollar, it is rated lower then high-end GPUs [9]. More recently, especially this last spring, application-specific integrated circuits (ASIC) have become popular. As more are produced and they become cheaper, they are expected to cause GPUs to become obsolete for mining purposes. While one of the better ones currently, the Avalon ASIC #1, computes 66 million hashes per second, much smaller than the Butterflylabs FPGA, it is much more efficient monetary wise, calculation 54.34 million hashes per second per dollar compared to Butterflylabs FGPA's 1.64 million hashes per second per dollar and the Model 5830 GPU's 3.28 million hashes per second per dollar. The ASIC does cost $1,300, but it can easily be seen to be the most efficient and cost effective of those analyzed[10]. The Bitcoin wiki page analyzes many different hardware and their comparisons can be found in more detail at source [8].

4 Cryptography used in the Bitcoin Protocol

Bitcoin's cryptographic security is based almost entirely on two core algorithms: a 256-bit verion of SHA-2 and the ECDSA. The strength of these algorithms, along with their relative speed to compute, allows Bitcoins to be mined and transfered efficiently, while still protecting them from being double spent or stolen from their rightful owners.

4.1 SHA-2

4.1.1 History of SHA-2

In 2005, a team at the Shangdong University in China published a paper detaining a collision attack on SHA-1, only requiring 2^{63} hash operations to find a collision, significantly smaller than the 2^{80} required in a brute force attack. Even with this major break in security of SHA-1, which Bruce Schneier noted as "put[ting] a bullet into SHA-1 as a hash function for digital signatures", the likelihood of pulling off a collision attack was still reasonably low. In 2012, Marc Stevens proposed a method of attack to break a single hash–at the low cost of $2.77 million USD. Despite the low likelihood and the vast resources required to pull off an attack, in 2010, the National Institute of Standards and Technology required government agencies in the U.S. to move to SHA-2.

Published in 2001, SHA-2, of which Bitcoin uses a 256-bit version known as SHA-256, is a one-way cryptographic hash function. SHA-2 was designed by the National Security Agency (NSA) as a successor to the SHA-1 hash function, differing greatly from the SHA-1 algorithm which it replaces.

4.1.2 SHA-2 Algorithm

The algorithm for implementing SHA-2 is similar in construction to that for implementing SHA-1. Note that $>>>$ denotes a right rotation, $>>$ denotes a right shift, $\oplus$ denotes an exclusive-OR operation, $\wedge$ denotes an AND operation, ! denotes a logical NOT operation, + denotes an addition (modulo 2^{32}).

1. Prepare the message by appending 1 and enough 0's to pad the length (in bits) to be congruent $448 \pmod{512}$.
2. Split the message into 512-bit chunks
3. For each chunk
   (a) Split the chunk into sixteen 32-bit words
   (b) Expand the words from sixteen 32-bit words to sixty-four 32-bit words, such that $w[i] = w[i-16]+((w[i-15] >>> 7) \oplus (w[i-15] >>> 18) \oplus (w[i-15] >> 3))+w[i-7]+((w[i-2] >>> 17) \oplus (w[i-2] >>> 19) \oplus (w[i-2] >> 10))$ for $16 \leq i \leq 63$.
   (c) Initialize $a..h$ to $h_0..h_7$.

(d) For $0 \leq i \leq 63$
   i. Set $temp = h + ((e >> 6) \oplus (e >> 11) \oplus (e >> 25) + ((e \wedge f) \oplus (!e \wedge g)) + k[i] + w[i]$, where $k[i]$ is the ith round constant.
   ii. Set $d = d + temp$
   iii. Set $temp = temp + ((a >>> 2) \oplus (a >>> 13) \oplus (a >>> 22)) + ((a \wedge (b \oplus c)) \oplus (b \wedge c))$
   iv. Set $h = g, g = f, f = e, e = d, d = c, c = b, b = a, a = temp$

(e) Set $h_0 = h_0 + a, h_1 = h_1 + b, ..., h_7 = h_7 + h$

4. Output $h_0h_1h_2h_3h_4h_5h_6h_7$ as the hash

4.1.3 Attacks on SHA-2

As of the time of writing, there are currently no known collision or preimage attacks on the full 64-step SHA-256. However, some potential preimage and collision attack methods have been proposed on a version of SHA-256 reduced to 41-steps. This preimage attack, proposed by *Sasaki, Wang, and Aoki*, still requires 2^{240} SHA-256 calculations – roughly 2×10^{53} more calculations than required to find a collision in SHA-1.

4.1.4 SHA-3

As SHA-2 was based on SHA-1, which has known attacks, it would theoretically be possible for an attack on SHA-1 to be extended to SHA-2. Even though an attack on the full versions of SHA-2 has not been found yet, the NIST opened a contest in 2007 to find an alternative dissimilar hashing method to SHA-2. After years of testing and analysis, in 2012, the NIST announced that Keccak (pronounced "ketchak") would become the new SHA-3 hash algorithm.

Keccak is a sponge-based cryptographic hashing function. Sponge constructions take in (or "absorb") data as a stream at a certain rate and then output (or "squeeze out") data at a separate rate. As a result, Keccak can be used for various sizes ranging from the "toy" Keccak-f[25], which uses 25-bits of state, to the suggested Keccak-f[1600], which uses 1600-bits of state. In addition, the developers of Keccak suggest that the algorithm should be secure against preimage and collision attacks, even when only using half the number of rounds (12 out of 24 rounds in the case of Keccak-f[1600]).

4.1.5 Usage of Cryptographic Hashing Functions in Bitcoin

Bitcoin uses SHA-256 as part of the block hashing algorithm and the proof-of-work mechanism. However, even though current implementations of Bitcoin use SHA-256, the general mechanism of Bitcoin could potentially use any cryptographic hashing algorithm, such as SHA-512 or Keccak/SHA-3. Such a switch of algorithm would be somewhat difficult, requiring all of the nodes in the Bitcoin network to upgrade to software versions using the new algorithm, but it would still be possible without significant changes.

4.2 ECDSA

5 Additional Considerations

From the technological standpoint discussed earlier in this paper, Bitcoin seems to be an excellent way to perform quick currency transactions between individuals, discretely or even anonymously. Since Bitcoin is so unlike any other currency seen before, there have been many concerns about its long term financial stability, as well as legal issues surrounding transfer of funds, especially from national and international tax orginizations, as well as the use of Bitcoins as payment for items on the black market.

5.1 Economic Concerns

5.1.1 Bitcoin Supply

The Bitcoin supply is controlled by a geometric function that releases Bitcoins slowly, so as to not create large fluctuations in the price, and to introduce stability in the currency. The expansion of the currency over time looks like this:

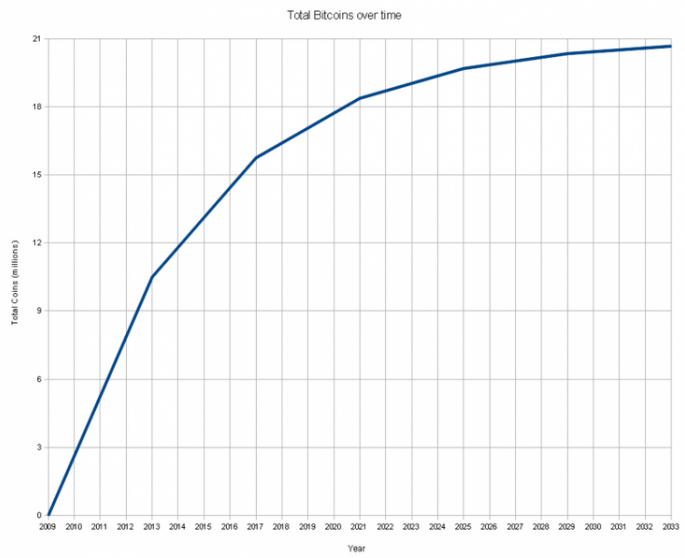

Figure 8: Bitcoins in circulation over time [11]

The nodes on the Bitcoin network change the difficulty of mining in order to keep inflation of the money supply from increasing too rapidly, thus devaluing the currency and potentially even making it collapse. The current value of Bitcoin is estimated at about a half billion USD [12].

5.1.2 Bitcoin Crashes

Though the nodes may be able to keep miners from devaluing the currency by creating too much of the product, Bitcoin traders can buy and sell the currency, creating value or devaluing it, depending on what they are doing with the currency. Note that this functions exactly the same as any other fiat currency (US Dollar, EU Euro, etc.), the difference being that the money supply cannot be arbitrarily changed by a central authority, since there is no central authority. Since the value is decided almost entirely by the holders of Bitcoins (who in the beginning were largely Bitcoin miners themselves), the currency is thus far more subject to fluctuations. Once Bitcoin gained popularity, and people began to infuse large amounts of money into it, problems began to arise, and eventually, Bitcoin experienced a monumental crash, where the price dropped from $266 USD to almost $105 USD in a few hours. This crash is shown below:

Figure 9: Bitcoin crash of April 10, 2013 [13]

As the total value of the Bitcoin market approaching one billion USD (driven in part by individuals seeking tax havens or starting to use Bitcoin as an investment), speculation that this was a bubble that was about to burst began circulating, only to be realized shortly after, resulting in a halving of the total value of the Bitcoin market.

5.2 Legal Concerns

5.2.1 In the world

Bitcoin gained much of it's current popularity due to the financial crisis in Cpyrus in late 2012/early 2013. The government of Cpyrus decided to take

money from bank accounts held in it's banks in order to repay it's debts, and many wealthy Russians (who used Cyrpus as a tax haven), cashed out of those banks and into Bitcoin, driving the price up significantly. In light of this, and the overall potential for abuse of digital currency (as a tax haven or as payment for illegal goods or services), many governments have begun to issue statements about virtual currencies, most notably the US Department of Tresury.

5.2.2 In the US

In March of 2013, the Department of the Treasury's Financial Crimes Enforcement Network (FinCEN) issued a memo that classified anyone who exchanged "virtual currencies" for physical currency (USD, GBP, etc.), would be subject to licensing and regulation[14]. While this doesn't currently apply to individual users of currencies such as Bitcoin, it does affect the administrators of the Bitcoin exchanges, and may result in tens of thousands of dollars in additional costs, reducing the number of exchanges in the US and pushing them to other countries that are less restrictive[15]. As the total value of Bitcoin continues to increase, and the potential tax evasion issues are dealt with, more regulation is likely to come, especially to individuals using the currency.

6 Summary

Bitcoin lies at the crossroads of cryptography and economics, and is an interesting case of the literal value of keeping secrets secret. Bitcoin's security lies in the strength of SHA-2 and the ECDSA, and has yet to be hacked in it's most recent versions. The weakness of Bitcoin lies in two things: 1) the idea that more of it's users are fundamentally good than bad (and that the bad ones can't rebuild the entire chain without the good ones creating more), and 2) the future regulation of currency by central authorities. While Bitcoin is crpytographically secure, it still has to prove itself in the free market, that is is a viable alternative currency, and that it will remain stable in the long run.

References

[1] Bitcoin Protocol Specification. [Online]. Available: https://en.bitcoin.it/wiki/Protocol_specification

[2] The Cryptography of Bitcoin. [Online]. Available: http://blog.ezyang.com/2011/06/the-cryptography-of-bitcoin/

[3] Bitcoin Paper. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[4] Bitcoin Vocabulary. [Online]. Available: http://bitcoin.org/en/vocabulary

[5] Bitcoin Addresses. [Online]. Available: https://en.bitcoin.it/wiki/Technical_background_of_Bitcoin_addresses

[6] Bitcoin Paper. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[7] Bitter to Better - How to Make Bitcoin a Better Currency. [Online]. Available: http://crypto.stanford.edu/ xb/fc12/bitcoin.pdf

[8] Bitcoin. [Online]. Available: https://en.bitcoin.it/wiki/Main_Page

[9] Butterflylabs products. [Online]. Available: http://www.butterflylabs.com/products

[10] Avalon Miner Power Usage. [Online]. Available: http://garzikrants.blogspot.com/2013/02/avalon-miner-power-usage.html

[11] Total Bitcoins over time. [Online]. Available: http://en.wikipedia.org/wiki/File:Total_bitcoins_over_time.png

[12] Magic the Gathering Online Exchange Chart. [Online]. Available: http://bitcoincharts.com/markets/mtgoxUSD.html

[13] Bitcoin Collapse. [Online]. Available: http://www.naturalnews.com/images/Bitcoin-Collapse-Bitcoinbullbear.jpg

[14] Application of FinCEN's Regulations to Personas Administering, Exchanging, or Using Virtual Currencies. [Online]. Available: http://www.fincen.gov/statutes_regs/guidance/pdf/FIN-2013-G001.pdf

[15] FINCEN: Bitcoin Users Not Regulated, Exchanges Are. [Online]. Available: http://bitcoinmagazine.com/fincen-bitcoin-users-not-regulated-exchanges-are/