Indian Institute of Technology
Kanpur

Department of Computer Science and Engineering

Advanced Computer Networks : CS625A

Bitcoin Research Report

Prepared By:

Sagar Chand 14579

Supervisor:

Professor Dheeraj Sanghi

Bitcoin Research Report

Sagar Chand, 14579

April 24, 2018

Abstract

Bitcoin has emerged as the most successful crypto- graphic currency in history. Launched in 2009, Bitcoin grew to comprise billions of dollars of economic value Currently its market capitalization is more than $150 Billion. Earlier this year it crossed Market cap of $200 Billion which is more than that of Goldman Sachs, Morgan Stanley, Master Card, IBM, GE, Tesla etc. Since then a lot have been discovered about this virtual currency like important properties of the system, discovered attacks and proposed promising alternatives. This report, apart from explaining about Bitcoin, also summarizes various aspects of it and the many related cryptocurrencies or 'altcoins'. Bitcoin's anonymity issues is also present in this report.

1 Introduction

Bitcoin is software-based online payment system described by Satoshi Nakamoto in 2008 and introduced as open-source software in 2009. Payments are recorded in a public ledger using its own unit of account (Bitcoin). It is a form of digital currency, created and held electronically. It can be used to buy things electronically and in that sense it is no different than conventional dollars. Bitcoin is commonly referred to as cryptocurrency and it can be divided into smaller unit called Satoshi (one hundred millionth of a BTC). According to Satoshi Nakamoto [1], It is a purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

2 Need for Bitcoin

Trusted Payment - It is trusted mode of payment

Electronic - So no forging of currency

No need for 3rd party - Unlike traditional online transaction method which requires one or more intermediary(eg banks) to verify that payment has been made, bitcoin does not require any intermediary.

No double Spending - There is tendency to cheat by giving same money to multiple people and hence double spending if there is no 3rd party to watch but this is not the case with bitcoin payments.

3 Important Cryptology Concepts

3.1 SHA 256

It takes the message that you're hashing, and it breaks it up into blocks that are 512 bits in size. The message isn't gonna be, in general, necessarily exactly a multiple of the block size, so we're going to add some padding at the end. And the padding is gonna consist of, at the end of the padding, a 64 bit length field, which is the length of the message in bits. And then before that, it's gonna consist of a one bit, followed by some number of zero bits. And you choose the number of zero bits so that this comes out exactly to the end of a block. So once you've padded the message so that its length is exactly a multiple of the 512 bit block size, you then chop it up into blocks, and you then execute this computation. You start with the 256 bit value called the IV. That's just a number that you look up in a standards document. And then take the IV and the first block of the message. You take those 768 total bits, and you run them through this special function, c, the compression function, and out comes 256 bits. You now take that with the next 512 bits of the message, run it through c again, and you keep going. Each iteration of c crunches in another 512 bit block of the message and mixes it in, sort of logically to the result. And when you get to the very end, you have consumed all of the blocks of the message plus the padding. The result is the hash, that's a 256 bit value. And it's easy to show that, if this function, c, this compression function is collision free, then this entire hash function will also be collision free.

3.2 Hash Pointer

a hash pointer is basically a simple thing, that we're going to take a pointer to where some information is stored. And we're going to together with the pointer store a cryptographic hash of the information. So whereas a regular pointer gives you a way to retrieve the information. A hash pointer is gonna let us ask to get the information back. It's also gonna let us verify that the information hasn't changed. So a hash pointer tells us where something is and what it's value was.

3.3 Merkle Tree

suppose we have a bunch of data blocks which we'll draw across the bottom down here. We're going to take consecutive pairs of these data blocks and for these two data blocks we're going to build a data structure here that has two hash pointers, one to each of these blocks, and similarly all the way across. We'll then go another level up and this block here will contain a hash pointer of these two children down here. And so on, all the way back up to the root of the tree. And then just like before we're going to remember just the hash pointer up here at the head of the tree. And we can then, if we want traverse down through the hash pointers to any point in the list. And we can make sure that the data hasn't been tampered with. Because just like I showed you with the block chain, if an adversary tampers with some block down here at the bottom with the data that will cause the hash pointer that's one level up to not match. So he'll have to tamper with that. And therefore, he'll have to tamper with the hash pointer one level up from there. And eventually he'll get up to the top, where he won't be able to tamper with the hash pointer that we've remembered. So again, any attempt to tamper with any piece of data across the bottom will be in short against, by just remembering the hash pointer at the top.

4 Digital Signature

a digital signature is supposed to be just like a signature on paper only in digital form. And what that means is this, what we want from signatures is two things. First, that just like an ideal paper signature, only you can make your signature, but anyone who sees your signature can verify that it's valid. And then the second thing you want is that the signature is tied to a particular document. So that somebody can't take your signature and snip it off one document and glue it onto the bottom of another one because the signature is not just a signature. It signifies your agreement or endorsement of a particular document.
It broadly has 3 functions:-
(sk, pk) := generateKeys(keySize)
sig := sign(sk, message)
isValid :=verify(pk, message, sig)
Here, sk is secret key(or private key) and pk is public. key

5 Bitcoin Transaction

We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership.

Transaction Input - Specify a previous transaction specifically. Hash pointer to transaction. Index to transaction in block. Signature.

Transaction Output - Value, Recipient, Script

6 Bitcoin Scripts[2]

Each transaction output doesn't just specify a simple public key, it actually specifies a script. There are two scripts - scriptPubKey and scriptSig.
Typically, the scriptPubKey specifies the hash of an ECDSA public key and a signature validation routine. This is called a "pay-to-pub-key-hash" transaction and the entire redeeming transaction must be signed using a key with the the specified hash. The vast majority of Bitcoin transactions are pay-to-pub-key-hash and the system is often described with this being the only possibility, although other transaction types are possible. The scripting language is an ad hoc, non-Turing-complete stack language with fewer than 200 commands called opcodes. They include support for cryptographic operations—e.g., hashing data and verifying signatures. Like the transaction format, the scripting language is only specified by its implementation in bitcoind. Transaction inputs refer to previous transactions by their transaction hash and the index of the output within that transaction's output array. They must also contain a code snippet which "redeems" that transaction output called the scriptSig. To successfully redeem a previous transaction, the scriptSig and scriptPubKey, must both execute successfully, one after the other, using the same stack. For pay-to-pub-key-hash transactions, the scriptSig is simply a complete public key (with the correct hash) and a signature.

7 Bitcoin Decentralization

7.1 Distributed Consensus

When protocol terminates, most nodes decide on some value. Value must be proposed by come node.
At any point in time all nodes have a sequence of block of transaction they have reached consensus on. Each node has set of outstanding transactions it has heard about. Outstanding transactions may be different for different nodes. Consensus is difficult because some nodes may crash, few of the nodes may be malicious or/and network might not be perfect due to latency, faults etc.

7.2 Hash Puzzle

To create a block one has to find nonce such that Hash of block is very small. As of August 2014, 10^{20} hashes are computed block to find the correct solution to the puzzle. This puzzle is difficult to compute. This is random puzzle so that person who spend more resources do not take away all the reward. This puzzle has been designed in such a way that rewards are given roughly in proportion to the amount of computation you have carried out.

8 Bitcoin Network

8.1 Bitcoin P2P Network

- Ad-hoc Protocol (runs on TCP port 8333)
- Ad-hoc network with random topology

- All nodes are equal
- New nodes can join any time
- Forget non responding nodes after 3h

8.2 Joining the Network[3]

Nodes are connected to each other in random fashion. New node only needs to know how to get to one node that is already on the network(called seed node). You send getaddr() to that node asking to tell all the peers it has. Node responds by giving addresses of the nodes it is connected to. Then you send same message to its peers and get addresses of more nodes. Iterate as many times as you want until you have list of peers to make connection with. Choose the ones you want to peer with.

8.3 Transaction and block propagation

One of the nodes hear about new transaction. It tehn propagates it. Nodes will add this new transaction into transaction pool. Every transaction is identified uniquely by its hash so each node can tell they have seen that hash before and do not need to forward it. Transaction should be forwarded only if transaction is valid in current block chain, script matches whitelist, node hasn't seen transaction before and it doesn't conflict with others I have relayed.

8.4 Race condition[4]

Transactions or blocks may conflict:-
Default behavior - Accept what you hear first.
Network position matters - If you are closer to most nodes that other, then it is more likely that your transaction will be added to the chain and you will get the mining rewards.
Miners may implement any other logic they feel like.

8.5 Propagation

Some propagating algorithm similar to that of multicast is used to announce new block i the network. Transactions are verified by running scripts, nodes verify block by computing hash. They also make sure that it starts with sufficient number of zeros to meet difficulty target. Apart from validating header, caching has value, nodes also have to validate every transaction in the block.
It takes a lot of time to reach nodes farther away. Around 30seconds for a 250KB block to reach 75% of the nodes.[3]

8.6 Validating nodes

There are only 5 to 10 Thousand fully validating nodes. Others are either lightweight nodes or they are temporary, ie they come and go. The fully validating nodes are permanently connected. They store entire block chain. They hear and forward every node/transaction. The main reason they are few in number is because of huge size of Bitcoin network which is around 150GB as of April 2018.

8.7 Unspent transaction output(UTO)[5]

It is stored in RAM to check if transaction is already there or not. Everything else can be stored in the disk.

8.8 Lightweight nodes

These are also called Simple Payment Verification Client. These are not fully Validating nodes. They don't sore every thing. They request transaction as needed to verify the incoming payment. They have to trust fully connecting nodes.

9 Storing and using Bitcoin

9.1 Simple Local Storage

To spend a bitcoin, you need to know some information about blockchain(public info) and secret signing key of owner. So we need a way to store the secret key. We can store in phone or computer, or write it down. But there is possibility that those things are damaged and hence our coins are lost forever.

9.2 Wallets[8]

They can be used to keep track of our coins. They use separate key for each coin we have. It benefits privacy as we look like separate owners. Wallets can do all the book keeping required and user need not know.
Now we do not want a lot of coins in our active wallet, so we keep extra coins in our offline storage. We do not want rest of the world to know that we have an offline storage. Hence forth we need different address each time we transfer money in that account. For that we use hierarchical key generation technique[9]. This is an address generating script, which generates a key in our offline storage and corresponding public key in our active wallet each time a coins are transferred to the offline wallet.

10 Bitcoin Mining

10.1 Task of Miners

They join the network, listen and validate the transactions.
They listen for new blocks, maintain blockchain and validate whenever a new block has been proposed.
They assemble a new valid block.
They find the nonce that make your block valid.

The task of miners is to find a nonce such that the hash of entire block has atleast 'd' zero's in its hast in front. Sometimes you won't find required hash, so you change the parameter in coinbase transaction. Coinbase transaction is where miners are actually mining new coins and claiming it for themselves. There is extra nonce parameter there. So we change its value by 1.

Difficulty of block is chosen every 2 weeks so that block is chosen every 10 minutes.

10.2 Mining Pools[6]

Mining is a random process. You don't know when you're gonna find the next block. It's a completely random search, and you could find your next block at any time. Variance is pretty high. Now the goal of a mining pool is that a group of miners will get together, form a pool. And they'll all attempt to mine a block, which pays that coinbase, the newly minted coins to the same recipient. That recipient is going to be called the pool manager. So no matter who actually finds the block, the pool manager will control the rewards. And then the pool manager will take that revenue and distribute it to all of the participants in the pool based on how much work each participant actually output. Of course, the pool manager will also probably take some kind of cut for their service of managing the pool.

10.3 Incentive

Whosoever node, which computes the block and gets its block accepted gets a block reward. It was 50BTC when it was started and since then it has been halving every 2 weeks. Thus at the end of 2140, there would be no rewards and 21million BTC in all. Henceforth a transaction fee can be charged, which would then be given by transacting parties and would be paid to the node that computes the accepted block.

11 Attacks

11.1 Goldfinger Attack[7]

If a group/mining pool can take over atleast 50% of the nodes in a block chain network, it can control the network. It is also known as 50% attack. That group can then double spend, change values and do whatever it feels like, thus decreasing the trust in that block chain and collapsing it. eg-CoiledCoin was destroyed by Eligius group as they saw it as a threat to bitcoin.

11.2 Sybil Attack[4]

Copies of nodes that a malicious adversary create to look like a lot of different participants. This way, he can pretend to have more number of nodes implying more votes thus implying more power over the network.

12 Limitation and Improvements

- 7 transaction per sec[10] - Each block is limited to a million bytes. And each transaction has to be at least 250 bytes, so if you divide through, and the fact that blocks are found every 10 minutes, you're left with about 7 transactions per second which is all that the Bitcoin network can handle.
- Puzzle can be broken in long term eg- by using rainbow table

- Hard forking is nearly impossible - All nodes have to upgrade, otherwise blockchain will split. In current world, many people do not see their nodes for a long period of time and hence it cannot update their nodes, so no major changes can take place in bitcoin block chain. Thus newer chains with extra features will prevail in future

13 Trying

I was trying to implement few applications of blockchain like Decentralized voting and smart contracts. Both of them are in their initial stages.

Acknowledgments

I thank my instructor, **Professor Dheeraj Sanghi** for motivating me and guiding me throughout the project.

References

[1] Satoshi Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System* 2008

[2] Bonneau J, Miller A, Clark J, Narayanan A, Kroll JA, Felten EW, *SoK: Research Perspectives and Challenges for Bitcoin and Cryptocurrencies* 2015

[3] d'Enginyeria de la Informaci i les Comunicacions Universitat Autonoma de Barcelona, *The Bitcoin P2P network* 2014

[4] Mauro Conti, Sandeep Kumar E, Chhagan Lal, Sushmita Ruj, *A Survey on Security and Privacy Issues of Bitcoin* 2017

[5] Khaled Baqer, Danny Yuxing Huang, Damon McCoy and Nicholas Weave, *Stressing Out: Bitcoin "Stress Testing"* 2016

[6] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden, *Incentive Compatibility of Bitcoin Mining Pool Reward Function* 2016

[7] Joshua A. Kroll, Ian C. Davey, and Edward W. Felten Princeton University, *The Economics of Bitcoin Mining, or Bitcoin in the Presence of Adversaries* 2013

[8] Möser M, Eyal I, Sirer EG, *Bitcoin Covenants* 2016

[9] Gutoski G, Stebila D, *Hierarchical deterministic Bitcoin wallets that tolerate key leakage* 2015

[10] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Sirer, Dawn Song and Roger Wattenhofer *On Scaling Decentralized Blockchains* 2016

[11] https://bitcoin.org/en/developer-guide