BLOCKCHAIN REFLECTIVE ESSAY

May 2019

Nidhi Kadkol
UT EID: nk9368
nidhikadkol@cs.utexas.edu

Contents

0.1 Overview

This paper describes my journey of getting to grips with blockchain technology - the technical details, implementation, and its use cases. I entered the course having no knowledge of what blockchain means, only having vaguely heard of it in relation to the bitcoin frenzy that everyone was in a couple of years ago. As the class progressed, I not only learnt about what this fascinating concept is, but I also got a chance to look at it from a social and philosophical viewpoint. This paper is organised according to the topics which made an impact on me that I learnt over the duration of this course, and thus start with basic concepts having further sections build up on previous ones. I have also included my views and what I felt about some topics. Reading this would give the reader an eye-lens to this course as it unfolded from my perspective.

0.2 The Blockchain Ideology

0.2.1 Cypherpunk-ism

The start of the course was devoted to understanding the motivation behind blockchain and its history. This goes way back to 1992-93 when the rise of the crypto-anarchy movement began. The internet was just beginning to be accepted by the world, Silicon Valley was steadily growing. Within this new-found age of free information and message passing, there was a group of people who were firm believers in the idea of total anonymity during communication and transactions of any sort. They believed that this could be achieved with the internet. They also were aware that the government would make efforts to prevent or slow this down from happening, as while privacy is what many people believe to be a right, it could also be abused by people for all kinds of evil purposes such as child trafficking, monetary fraud, and so on.

0.2.2 The need for privacy

A line that really hit home when reading Eric Hughes' "A Cypherpunk's Manifesto" was the clear distinction between secrecy and privacy, and that evil

people require secrecy, while crypto-anarchists demand privacy. He says that "A private matter is something one doesn't want the whole world to know, but a secret matter is something one doesn't want anybody to know. Privacy is the power to selectively reveal oneself to the world."

0.2.3 The need for cryptography

This desire for privacy leads to a natural requirement for anonymous systems and encryption of messages. Messages and information that is sensitive in nature, i.e. should not be publicly available, must be "locked" or encrypted so that only the parties involved will be able to "unlock" or decrypt the message to be able to read it. This is the foundation of the blockchain ideology.

0.2.4 The bitcoin ideology

The first product that used blockchain was bitcoin, and so it is natural to talk about the bitcoin ideology in the history of blockchain. Bitcoin was introduced to the world on Jan 3, 2009 by Satoshi Nakomoto (whose actual identity remains unknown). Bitcoin allows the transfer of money among people unconstrained by geographical borders, outside government control, and in a way that ensures privacy. The New York Times called his white-paper a document that was fundamentally political in nature. Bitcoin was a manifestation of the crypto-anarchist movement and cypherpunk-ism - it was seen as a tool to *disrupt* existing deep-rooted mechanisms of institutions like banks.

0.3 Cryptography

In this section, I will talk about some of the concepts of cryptography that are essential to understanding how blockchain works.

0.3.1 Symmetric Key cryptography

In this method, if Bob wants to send a message to Alice but not make the message viewable to anyone else, he encrypts the message with a key. He then sends the encrypted message to Alice, who can decrypt it with the same key. However, this kind of message sending is very susceptible to attacks as anyone who gets hold of the key will be able to intercept messages or even impersonate someone and send false messages. This is particularly evident when initially Bob has to share the key with Alice in order for both of them to be able to communicate via encrypted messages. In that case, someone who intercepts his message which has the key now gains access to the key.

Bob can get around this by encrypting the message which has the key by using a different key. But then again, he has to send this different key to Alice first, which leads to the same vulnerabilities!

0.3.2 Public Key cryptography

In this kind of encryption mechanism, everyone possesses a public key and a private key. The public keys are made known to everyone. Any message can be encrypted with a public key and decrypted only with the corresponding private key. So now if Bob wants to send a message to Alice in a secure manner, he encrypts the message with Alice's public key. Even if someone gets access to the message, he or she cannot read it as they do not have access to Alice's private key. Thus, this system is far more robust to hacking attempts than the one described in the sub-section above.

0.3.3 Computational efficiency

In class we learnt that using symmetric keys is fast and convenient, but susceptible to attacks. Asymmetric keys are safer, but computationally intensive.

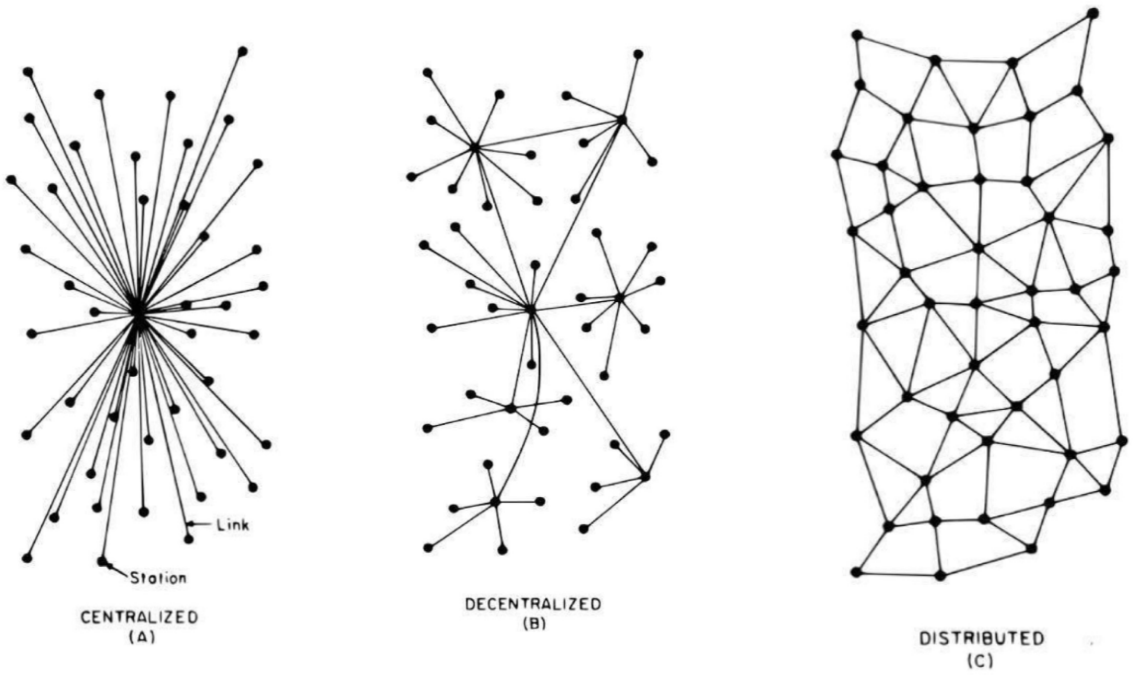
A hybrid approach combines the best of both worlds and uses the asymmetric approach to encrypt and send the symmetric key, after which messages can be exchanged efficiently using the symmetric key. This idea is elegant and simple and personally I was very impressed with it!

0.4 Blockchain Architecture

Much of my understanding of the overall architecture of how blockchain works came from the lab demonstration we had in our class by 2 people from a company called Consensys based on Ethereum.

0.4.1 Network topology

It is first important to understand how nodes can be distributed in a network. There are 3 kinds of network distributions that are relevant in the context of understanding blockchain, as shown in the figure below.



The first is **centralized** - here one node or organization is central to all the movement of data and information across the network. Every node must go though the central node in order to communicate with any other node.

The second is **decentralized** - here there are multiple nodes, or a group of nodes which function as centers through which communications flow. All the nodes form smaller clusters and each cluster has a center which is like the master node of that cluster.

The third, and most interesting one is **distributed** - here there is no central node - rather there is a huge mesh network where no node is the leader, and if one node fails, it is likely that the nodes which were connected to this node will have other paths or nodes they are connected to through which they can re-route information flows.

0.4.2 Hyperledgers

This is a key aspect to understand in blockchain - it is the basis of what kind of information is stored and how it is stored on the blockchain.

A simple ledger is a database used in accounting - which keeps a record of the transactions that occurred between people. The properties of a ledger are that it should be centralized, there must be data integrity, the ability to retrieve records must be quick, and there should be accessibility. Everyone turns to this centralized ledger to access information.

In a distributed ledger system, the ledger is not centralized. Rather, each person on the network has a copy of the ledger. In order to maintain data integrity and consistency, all the peers on the network must agree on the state.

0.4.3 Blocks

Blockchain is a type of distributed ledger which uses blocks to maintain the shared information and state. A block includes a list of transactions, the hash of the previous block, and a nonce. Each block contains a list of transactions, so all the blocks together would contain a complete list of transactions from the beginning of time till the present minute. So how do peers agree on the state when the transactions are stored in the form of a blockchain? If a new transaction is made, each peer adds that transaction to a block on their copy of the block. Suppose a set of transactions has occurred and now a new block needs to be added. If someone just decided to add the block, there is no guarantee that there is a genuine list of transactions on that block. So for this, we turn to the proof-of-work mechanism.

0.4.4 Proof of work

Proof-of-work is a guessing game with answers that are hard to find but easy to verify. In bitcoin, this game boils down to finding a nonce (which is a number) such that the hash of the (nonce + the previous block hash + the list of current transactions) has a minimum number of zeros. This minimum number is fixed to be the difficulty of the guessing game, or the proof of work. More number of leading zeros required means it is harder to correctly guess the solution, i.e. the expected number of brute-force tries to get a valid solution is higher. If a miner does find such a nonce, then he adds it to the block and calculates its hash. He then broadcasts the solution to the network along with the hash of the block. The participants on the network verify his solution, i.e. they check if the nonce that he has broadcasted produces the same hash value as he has claimed when they add it to their copy of the block with its transactions. If it does, then they accept this solution. If a majority of the participants accept this solution, then it gets added to the chain.

It is useful to think about why proof-of-work is required. Why do the miners have to play this guessing game? Can't they just find the hash of (the previous block hash + the list of current transactions) and broadcast that on the network, and the other miners verify it? Well, in this case it is much easier to broadcast fake transactions. Also, double spending becomes easier. If adding new blocks is an energy-intensive process, then malicious miners will think twice before attempting to do so as they have to expend resources and then rely on the slim (mostly impossible) chance that they still get a correct hash or the chance that they still get a wrong hash and it gets accepted (which is impossible).

Another aspect I really enjoyed discussing and learning in class was the 51% attack - that is, since the security of blockchain lies in the majority voting of the entities involved, what happens if a majority of the voters are against the system? They can collaborate and vote for a fraudulent block to be placed in the chain? While this looks simple on paper, it is extremely difficult as the probability of a fake block having the same hash value as a true block is astronomically low. And the computational power required to out-race the rest of the systems on the network and place this block also makes it much harder. In essence, such an attack is almost impossible, and this is by the design of blockchain.

0.5 Hashing

In this section, I will talk about the many things I learnt about hash functions and how it is relevant to blockchain systems.

Hashing means taking an input of any length and producing an output of fixed length. Hashing is **not** the same as encryption, i.e. We cannot "decrypt" the hash to get back the input. In other words, a hash is a one-way function. Hashing in blockchain is done for message authentication, digital signature verification, and so on. On the blockchain, instead of storing all the raw data we store the hash of the data instead - which also makes it easier to compare data on the blockchain, since comparing hashes is much faster than comparing huge amounts of data one bit at a time, and this also leads to memory efficiency. An ideal hash function is one in which the fastest way to compute a pre-image is through a brute-force attack, i.e. go through all of the possible inputs.

0.5.1 Properties

A cryptographic hash function needs to have certain properties in order for it to be considered secure. They are:

- **Deterministic** - that is, it always gives the same output for a particular input, no matter how many times that particular input is given or when it is given.
- **Fast** - Quick computation of the hash output from the input must be possible.
- **Small changes in the input result in huge changes in the output**.
- **Pre-image resistance** - This means that suppose you have a hash function f, and you're given a hash output y, it is difficult to find the input x such that $f(x) = y$.
- **Collision resistance** - It is computationally difficult to find any 2 distinct inputs x and x' such that $f(x) = f(x')$.

0.5.2 Use Cases

The main use case of hashing algorithms is for data verification. This comes up in a number of scenarios. For instance, digital signatures on contracts are done with a person's private key. Once a person signs a document, he can calculate the hash of the document along with his signature. In the future, if someone claims that he has signed a document with different terms and conditions, he could simply take the hash of that document and verify if it is equal to the hash he had computed earlier of the original document he had signed. If it differs, it means the document has been tampered.

It is also used for efficient storage of data on the blockchain. Instead of storing the huge amounts of raw data on the blockchain, we could simply compute the hashes of different sets of data and store that on the blockchain. Furthermore, if the number of hashes becomes too large, we could concatenate all the hashes together and find the hash of all of these to replace them with a single hash. This can continue in a Merkle tree fashion. From the deterministic and quick computation properties of hashing algorithms, we can do this quickly as well as verify the contents efficiently.

0.5.3 Collision attack - use case

What happens if a hashing algorithm is not collision-resistant? That is, it is feasible to find 2 inputs that give the same hash output? One serious repercussion is that digital signatures can no longer be trusted. Suppose I want to trick someone into signing a fraudulent contract. I prepare a fair contract M and a fraudulent contract M'. I can then generate many variations of M without changing its core content or meaning by doing things such as changing some spaces, punctuations, inserting new lines or removing some, and so on. Meanwhile I can do the same thing to M' and create many variations of M'. And now I can generate hashes of all these variations until I find a version of the original contract and a version of the fraudulent contract which have the same hash. I then present the fair contract to the person for singing, and then I take the signature and attach it to the fraudulent contract. This signature proves that the person has signed the fraudulent contract, and the resulting hash does not change which leaves me incapable of being suspected.

0.6 ECC and Brave Browser

In this last, sort of miscellaneous section I want to talk about two more things I learnt that I found pretty interesting. One of them was a guest lecture on Elliptic Curve Cryptography. I found the Elliptic Curve Cryptography talk really cool - I liked the fact that a 256 bit ECC key is as secure as a 3072 bit RSA key. ECC also makes a good random number generator. The cons of ECC are that it is resource heavy, has pre-set public parameters, and it is possible to hide back-doors. I also learnt about Kerchoff's principle which is that the best security is secure even when every piece (except the key) is publicly available. I never thought that we could use elliptic graphs for something like encryption - it made me appreciate the potential and wide use-cases of algebraic mathematics in general. I also learned that the industry standard for blockchain is the SECP 256 K1, and it is used by bitcoin, litecoin, ethereum, and so on. It is a form of ECC where the parameters are particular values, like A = 0, B = 7, and P and G are really big hexadecimal numbers. We also talked about quantum computers, which make use of Qubits. Qubits are an extension to the standard binary or 0-1 bits. They can also have a superposition of 0 and 1. We need 2000 qubits to break RSA - and just for perspective, the IBM Q System one has 20 qubits.

The second thing I learnt about is the Brave browser which uses the BAT or the Basic Attention Token. It puts the consumers and the advertisers first as opposed to mainstream browsers where the company takes almost all of the ad dollars. I even installed the Brave browser on my laptop and am considering making a complete switch to this browser from Chrome!

0.7 Conclusion

I have learnt so many things in this class in the most enjoyable way possible. I now have an understanding of how the blockchain architecture works, and I am also grateful for the discussions we had in class about the social, political, and philosophical aspects of this technology. It is rare to find a course which explores both the technical and humanities aspect of a movement, and the easy-going manner in which our professor, Dr. Lance Hayden conducted the class took off any kind of stress I would have originally had about this course and

definitely made it a class to look forward to every Friday afternoon. My journey with blockchain has been an informative one, coupled with introspection about values and principles of society as well as deep thinking about certain aspects of the architecture such as proof-of-work and 51% attacks. I look forward to actively following the future progress of blockchain.