In this ever-changing world of technology, we are getting to stages where payments method evolved from simply exchanging goods when in demand to now where payments are made in virtual cash. Recently developments of new types of virtual cash have evolved into safe ways to make payments online. By exchanging virtual tokens and cash instead of real life cash to make payments. However, the big problem with these types of payments is the trouble of problem of double spending.

**Double Spend Problem**

Double spending in simple terms is when you in effect spend already used cash. An example of double spending in real life is when I have 5 pounds in the debit card, I then use the same 5 pound note concurrently in two different shops. I spend the same amount in both shops but still have the 5 pounds or the 5 pounds is with one of the two. This may appear two both shops as a successful transaction however I have only 5 pounds in my account, someone is going to lose out.

Banks can keep track of this spending, by keeping track of each transactions and balance of your bank account. However, it is very inefficient as it is a centralized network and is a single point of failure[**First Book**]. It is not a great method in terms of speed and reliability, what happens if the server is down? No one can make payments. This is not a good solution. In this paper I want to go through 3 different cryptocurrencies, I feel that some banks could benefit from these types of cryptocurrencies, Why? It is because the elegant solutions these types of transactions have and the security they can offer Is exceptional. However, this paper will focus more on the technical aspects of Cryptocurrencies.

The Market is full of different types of Cryptocurrencies, in this paper I will focus on Bitcoin, Ripple and Titiam. These are interesting as they are 3 good cryptocurrencies in terms of success but have different ways of going around achieving success. This is what I will focus on the types of Technical aspects these have types of cryptocurrencies have. I want to compare to types of blockchains they use, and the other technical aspects they all have which make them unique.

**Byzantine Failures and Faulty Nodes**

Trust in a distributed network is hard to achieve. Whenever you depend on more than 1 system it can be very hard to get right. If someone hacks/ attacks one of systems, It can affect nearly every other node on the system. Especially when other systems may depend on outcomes of other systems such as consensus algorithms.

Achieving Byzantine fault tolerance is hard in a distributed network, it’s the process of functioning as intended in the presence of arbitrary component faults [**http://sce.uhcl.edu/goodwin/Ceng5334/downLoads/byzantine.pdf**]. Arbitrary faults are not normal faults, normal faults would be defined as faults such cause crashes ect. The way these faults occure is by processing requests/instructions in a incorrect way [**http://sce.uhcl.edu/goodwin/Ceng5334/downLoads/byzantine.pdf**]. Incorrect here will be defined as when the behavior deviates from specification.

These can be in the form of attacks or someone making incorrect outputs occur. This means that algorithms must cope with these kinds of failures, bitcoin Ethereum and ripple need to find ways to stop these attacks from occurring. They all have very elegant ways to stop this from happening. Some are overkill like bitcoin’s solution, but some are less costly.

Byzantine problem occurs in a few ways, most commonly Omission Failure which is simply the node not responding or failing to process a request.

Execution failure – is simply data corruption or responding to requests incorrectly [**http://sce.uhcl.edu/goodwin/Ceng5334/downLoads/byzantine.pdf**].

The problem originates when a general wants to agree on a time to start a attack on another city. Problem is doing this in the presence of traitors which would be fault machines in this case. The general will issue out a binary output either attack or retreat.

Voter systems tend to use these kind of approach, they can use median values, average values even mean values to decide which value to go with [**https://arxiv.org/ftp/arxiv/papers/1605/1605.03771.pdf**]. They usually rely on replicated components with a central voter. However, in binary output’s one simple change or fault component can sway the results of a medium based voter causing the voter to work with erroneous values.

The general solution is that you should have 3m + 1 processors to deal with faults? This tells us we need at least 3 times more normal processors than faulty ones. A lot of redundancy and very expensive! Also, a round of messages must be exchanged with a known drift time.

We will see how the cryptocurrencies deal with these types of faults. In a network such as these, it is very easy to become susceptible to these sorts of failures.

**Ethereum**

**Bitcoin**

Another solution to a peer-to-peer networking system. Again, the famous paper released in 2009 talks about the double spending problem with a peer-to-peer system.

Bitcoin claims to solve this problem by using hashing and proof-of-work. Bitcoins involved ordering of transactions in the system. Transactions sent in the bitcoin network means that the order in which you received a message will not be guaranteed that it will have been created at that time!

Bitcoin in the background uses public Key Encryption which seems to be central to all the cryptocurrencies. So much so that I would say it is what makes a cryptocurrency.

**Public Key Encryption and Digital Signature.**

Bitcoin uses a combination of private/ public key encryption and a digital signature to ensure identity uniqueness. Advantages of this method is that it provides very simple key distribution, it is easy to generate, public keys don’t need to be registered on a server. The public keys and the private keys mean that registration is simple and easy. From this private key a public key can be used to decode a message. A private key can be used to encode a message providing a signature. [**https://www.cgi.com/files/white-papers/cgi\_whpr\_35\_pki\_e.pdf**]

The workings of public/ private key encryption has the following behaviour. A public-key and private key pairing is generated, the public key is made public and is widely distributed. The private key is something to keep under wraps, almost like a password to your bitcoin public address. Lose your private key means losing your funds. As the private key gives access to your funds.

It is set up like this so only you, the private key holder can access a given account and can encode specific message. If a private key has encoded a message then if you have a public key for that pair, you know that the sender must be the private key holder, as only the private key holder can generate this message. Under the hood this is what bitcoin uses to prove that a signature is generated by that public/private key pairing[**https://www.cgi.com/files/white-papers/cgi\_whpr\_35\_pki\_e.pdf**]. Below is an example of Bob sending a digital signature to Alice.

Bob

Alice

Alice sends message along with her Public Key

Bob decrypts message with public key, hence proving that the message is from Alice

**Figure 1: Digital certificate cryptography public/private key encryption**

UTXO’s

Bitcoin uses this type of public key/ private key base to ensure owners have really sent a transaction, but to also to ensure that coins are passed to new owners. The way public key is generated is from elliptic curve maths, ensuring that you cannot just reverse engineer the public key to get the private key [**https://www.cgi.com/files/white-papers/cgi\_whpr\_35\_pki\_e.pdf**].

A question may be asked that if a private key is just a random output, how is it not possible that I could get someone else’s private key thus losing your funds! However this is very unlikely, if you think about the amount of atoms in the world which are around 10^50, compared to the amount of possible bitcoin addresses 10^48 [**https://s3.amazonaws.com/files.douglas.stebila.ca/files/research/presentations/20140520-Bitcoin.pdf**]. We should not see any changes in the foreseeable future.

**Ownership Chain.**

The ownership chain in Bitcoin is required to ensure coins are passed to rightful owners, but also they are not forged and sent unwillingly. This is where the cryptography is very useful, by providing a way to prove that a coin way there’s and then passing it on the next owner. It does this by first “unlocking” the funds with the previous owner private key and using their public key to verify their digital signature then digitally signing the hash. This in turn hands the coin to the next account [**https://bitcoin.org/bitcoin.pdf** ].

**Ledger Consensus**

So far In this paper, what we have seen Bitcoin providing security by signatures. There is no way in which someone can forge someone’s signature unless they have their private key [**https://www.cgi.com/files/white-papers/cgi\_whpr\_35\_pki\_e.pdf**]. However, in this distributed network we as mentioned before, double spending is a big problem in a distributed network. We need to define a order of which transactions are made, ensuring that transactions are verified but also coins are not double spent or changed later on.

If Bitcoin were to say trust the timestamps on the message, well it could be that the user tampered with the timestamp to achieve some fraudulent/ illegal activity.

Another example would be, say I have 5 Bitcoins and I want to send someone 5 Bitcoins, I release the funds, and this shows on my current ledger, however say I made a copy of this ledger before I sent this to the user. If I then tried to spend by copy the network and all nodes should be able to tell the difference between the faulty node/ transactions or attacker, and the real value.

From: Jordan To: Alice Amount: 5 Message: m1

From: Jordan To: Chelsea Amount: 5 Message: m2

**Figure 2 Double Spending in distributed System**

Cryptocurrencies need to try in effect, not try find attacker but just not make it possible for them to do so, this is done by using Blockchain using majority ledger consensus. Meaning the majority ledger wins in terms of votes if they have more.

**SHA256 Proof Of Work**

This is where all of bitcoin is centred and why it is infeasible for someone to try sway the consensus votes. Proof of work means that nodes must generate work to get a block of transactions approved. Every node but execute a series of math calculations, what does this mean overall. That a single attacker must somehow outpace the everyone else for their transactions to be accepted [https://bitcoin.org/bitcoin.pdf]. It is perfectly acceptable to think this can happen however we will see later it is very unlikely.

Bitcoin chooses to use SHA256 this is because it is a cryptographic hash function, it has irreversibility which is the key factor in this. What Bitcoin aims to do with this hash is to ensure that a decent amount of work has been done to pass of this single transaction.

The work done is in effect solving a block of transactions by finding an input for a given output [**https://www.cs.upc.edu/~mjserna/docencia/grauA/P17/Crypto.pdf**]. this means finding an input that could be one of T/2^256 different possibilities which is amazingly large. Therefore, it takes up so much power for miners to solve this. There are different types of algorithms about these algorithms, but this is out of scope for this paper.

The calculation they must solve is very specific a hash by doing **SHA(**transactions, prev block, nonce**) <** t. Where t is a target value, the only thing that changes here is the nonce. The only way that someone can try get the target from a given hash is just by guessing. The target is that the hash must start with a certain number of zeros [**https://www.cs.upc.edu/~mjserna/docencia/grauA/P17/Crypto.pdf**].

The workings of SHA256 is a fascinating one, and really should make people feel happy when someone pitches bitcoin to them.

//DISCARDED BLOCKS ARE NOT GONE FOREVER.

Cryptographic – avoid collisions and to ensure no reversibility.

Which could mean fraud can be introduced in the bitcoin system. What could happen is that for example Alice wants to buy a product from bob, Alice sends Bob 5 bit coins and lies about timestamp. Alice then sends 5 bitcoins back to herself. What this means is that nodes on the network could receive the double spend before bob. Opening security flaws and untrust created in the network [**http://www.imponderablethings.com/2013/07/how-bitcoin-works-under-hood.html**].

What does this mean? The problem of defining a total order on a distributed system, this is arguably one of the hardest parts of achieving a system which relies on time. There are so many variables that can affect a distributed system even the crystals in the CPU’s themselves [**https://amturing.acm.org/p558-lamport.pdf**]. There needs to be rules which can give a total order given the use of time. If you can come up with an algorithm that can give total order then you can have a list of processes, or transactions in this case in which you can be sure than one transaction is ordered before another.

For Bitcoin a very sophisticated algorithm is put in place, called the blockchain. Giving a total order where transaction have taken place. The aim to define an order in which all transactions are created.

**Ripple**

The first Cryptocurrency is Ripple, Ripple is in my opinion looking like it could be one of the hottest new crypto currencies on the market. Ripple has an adopted a new technique to cope with double spending inside of the network. The focused paper [**Ripple**], which is the consensus whitepaper released for ripple, involving David Schwartz the main cryptographer of ripple explains how Ripple works. The paper shows the inner workings and the technical aspects of these.

Rules are defined in the following way; all nodes must oblige to these rules if they are able to pick up fraudulent transactions and try eliminating them from play.

C1, every nonfaulty node decides in limited time. C2, all non-faulty nodes reach same decision. C3, 0 and 1 are possible values from non-faulty nodes. non-faulty is defined as a node that produces a wrong error due to corruption.

In this system there are problems are a possibility that a system may be stuck on a problem, this could be in the form of a consensus on a node being stuck on a 50/50 decision. The system wants convergence and wants everyone to agree, therefore the probability is boosted in the system to make it for example a 60/40 to eventually converge to everyone agreeing [**Ripple**]. This is due from the work of Michael J fisher [**Impossibility of distributed Consensus with one faulty process**] , it concludes that when you have a asynchronous system of processes you may have some which are unreliable, in which even when one fault process occurs it is possible to have non termination.

The Ripple Protocol claims to support up to (n - 1)/5 failures, which is actually good performance. However, there are other factors which do make it better than just the formula mentioned above.

I will explain overall how the algorithm works in practice, it is not too complicated, and it is easy to see the benefits this system can offer.

What generally happens in the network is that this algorithm happens every couple of seconds, by all nodes. Transactions that are included will be a set of transactions, transactions that might have failed for some reason may be included in current rounds. When a final decision is made this open ledger becomes a closed ledger meaning the last known transaction which was valid.

Looking at this we cans see, during this 5 second transaction you can see transactions are very quickly resolved. Also when wanting to access for example somebodies last transactions, well closed ledger is there with a O(1) constant time access due to it just sat there [**Ripple**].

During each stage a set of transactions which appear valid is put forward by each node. What defines Valid? At this stage it is just to check that ripple transactions formation is being properly confirmed too [**Ripple].**

Once all transactions are made public, then nodes only vote on the transactions of servers which are in its UNL. If a transaction is not passed onto next stage, then what happens is the failing transaction is either put onto the next consensus candidate set.

If passed the transaction is therefore known as the last closed ledger. 80% is the threshold that must be agreed on for the open ledger to be transferred into the last closed ledger. This means that this will always work if 80% of the servers UNL agree. This also means that only 20% can be faulty in a UNL list, if it is more than it could be possible in the network to suffer from failures.

Correctness can be maintained in this type of network only if f<=(n - 1)/5 [**Ripple**]. However, security can still be assured in that fraudulent transactions will not be processed. This is because you now know that you cannot trust for example 30% of the nodes, therefore no transaction should be processed.

Even when a transaction which is illegal or fraudulent, a double spend transaction cannot be processed due to if one transaction has been processed then the other cannot be. Due to all nodes on the network acting on the same net of transactions therefore they will pick this error up. Rules or verifiers as they are called are applied to ensure that rules are in place when a consensus passes, verifications are applied ensuring that certain transactions are halted.

Agreement also is the action in which all no faulty nodes come to the same consensus, there is a problem however that when nodes do not have the exact same nodes in their list that they come to an agreement. This is since the nodes given in an UNL are not enough to satisfy the strong correctness given by 20% of faulty nodes. As there may be 20% of faulty nodes the server may not be aware of. Therefore, there must be a bound on prove agreement so that agreement is fair. It will always work if the intersection is larger than bound required to guarantee and agreement.

Means that there will always be a termination in the agreement stage. Must be met for the final UNL. Meaning high latency nodes are not included this is because each stage is quick and nodes that take a long time to reply makes it difficult to execute everything in time. So these are not included in the UNL. Ensuring nodes which which take longer that preset bound b are not included.