Instituto Tecnológico Autónomo de México

# Blockchain Seminar

**Talks by:**

Sergio Rajsbaum

Amir Herzberg

Amr El Abbadi

Maurice Herlihy

Aniket Kate

Mexico City March 2019

# Contents


**1 Introduction** **3**
- 1.1 Consensus in Distributed Systems . . . 3
- 1.2 The Blockchain . . . 4
  - 1.2.1 Traditional Banking System vs. Blockchain . . . 4
  - 1.2.2 Structure . . . 5
  - 1.2.3 Mining and Proof of Work . . . 5
  - 1.2.4 Properties . . . 6
  - 1.2.5 Applications . . . 6
  - 1.2.6 Regulation . . . 6
  - 1.2.7 Other Issues . . . 6

**2 Cryptography and Security** **7**
- 2.1 Hash functions . . . 7
  - 2.1.1 Hash functions properties . . . 8
- 2.2 One Way Functions . . . 9
  - 2.2.1 Difficulty to find $x$ . . . 10
  - 2.2.2 Second preimage resistance. . . . 10
- 2.3 Hash Block . . . 11
- 2.4 Blockchain . . . 12

**3 Permissioned Blockchains** **15**
- 3.1 Practical Byzantine Fault Tolerance (PBFT) . . . 16
- 3.2 Permissioned Blockchain Systems . . . 17
  - 3.2.1 Tendermint . . . 17
  - 3.2.2 Hyperledger Fabric . . . 18
  - 3.2.3 ParBlockchain . . . 18

**4 Alternative Design for Blockchains** **21**
- 4.1 Bitcoin NG . . . 22
- 4.2 ByzCoin . . . 22
- 4.3 Elastico . . . 23

**5 Solidity Pitfalls and Hazards** **25**
5.1 Re-Entrancy Attack . . . 25
5.2 Silent-Overflow Attack . . . 26
5.3 Unexpected Ether Attack . . . 27
5.4 DelegateCall Attack . . . 28

**6 Blockchain for Supply Chain and IoT** **29**
6.1 Blockchain technology . . . 29
6.1.1 Core industrial Architecture . . . 29
6.1.2 Current Applications . . . 29
6.2 Traditional Supply Chains . . . 30
6.3 Blockchain for Automotive Supply chains . . . 30
6.4 Blockchain for AI Marketplace . . . 31
6.5 Blockchain for Digital Supply chains . . . 31

**7 IOU Credit Networks** **33**
7.1 IOU Credit networks . . . 33
7.1.1 Tracing paths . . . 34
7.1.2 Transactions . . . 34
7.2 Sybil attacks . . . 35
7.2.1 Tolerance to Sybil attacks . . . 35
7.3 Proof of work . . . 36
7.4 IOU in the future . . . 36

**8 Atomic Cross-Chain Swaps** **37**
8.1 Atomic cross-chain swaps . . . 37
8.1.1 What could go wrong? . . . 38
8.2 Atomic Swap protocols . . . 39
8.2.1 Swap Diagraphs . . . 39

**9 Blockchain Scalability via Layer 2 Protocols** **41**
9.1 Scalability in Layer 1 . . . 41
9.1.1 Bitcoin in Layer 1 . . . 42
9.1.2 Scalability in layer 1 drawbacks. . . . 43
9.2 Scalability in Layer 2 . . . 43
9.2.1 Bitcoin in Layer 2. Lightning. . . . 44

**10 Blockchain Privacy: Challenges, Solutions and Unresolved Issues** **45**
10.1 Privacy issues . . . 45
10.2 Key approaches for blockchain privacy . . . 46
10.2.1 Mixing (Coinjoin mixing and third-party Mixing) . . . 47
10.2.2 Ring Signatures . . . 47
10.2.3 Stealth Addresses . . . 47
10.3 Conclusions . . . 47

**11 Wallets and Custody** **49**
11.0.1 Types of wallets . . . . . 49
11.0.2 Custody Rule in the USA . . . . . 51

**12 Open Problems Innovations and Challenges** **53**
12.0.1 Attacks inside the blockchain . . . . . 53
12.0.2 The Lightning Network . . . . . 54
12.0.3 Challenges and conclusions . . . . . 54

# Chapter 1

# Introduction

Talks by: Sergio Rajsbaum, Amr El Abbadi, Maurice Herlihy
Chapter by: Andrea Vargas and Rodrigo Medina

The popularity of the term *Blockchain* has increased dramatically in recent years. But, what is this technology and why has almost every newspaper already reported on it? In this work we present a summary of the Blockchains Seminar talks that took place at ITAM in March, 2019.

## 1.1 Consensus in Distributed Systems

The first paper on distributed computing systems was published in 1974. The main idea was to run a Universal Turing Machine (UTM) in a distributed architecture over several machines, this UTM should also be universally accessible and fault tolerant. But for this to happen, some problems have to be solved.

First of all, what would happen if some of the nodes in the system do not agree? Who must decide? Should every process wait to reach a consensus to guarantee consistency? Then, what time model are we working in? Is it synchronous? How do nodes communicate? These and other issues such as setting reliable broadcast algorithms, establishing the kinds of failures that the system should tolerate or defining how to handle backups represent a challenge when designing the whole model.

But the goal is clear, a distributed system must function as if it were a single system, being indistinguishable in its behaviour from one who runs on a single computer. Because of this, all the nodes in the system will have to set rules in order to reach a consensus in any case of discrepancies.

A consensus protocol most obey 3 principles:

1. Termination - every process decides eventually

2. Validity - the value chosen by a process needs to be a value that some process proposed.

3. Agreement - every correct process decides the same value

Unfortunately, in 1985 Michael J. Fischer, Nancy Lynch, and Mike Paterson proved that in a fully asynchronous system there is no consensus solution that can tolerate one or more crash failures. This result is called the FLP impossibility proof.[1] This does not mean that consensus can not be reached, this only states that it cannot be guaranteed.

After the FLP, some protocols were designed to try to reach consensus almost every time or with a high probability.

## 1.2 The Blockchain

In October 2008, in the middle of an economic crisis and with an atmosphere of distrust in the financial system, Satoshi Nakamoto (identity unknown) published a paper proposing a new coin called *Bitcoin* and with it introduced a new distributed, transparent, traceable and tamper-proof ledger in which every bitcoin transaction is registered. This ledger was called ***Blockchain*** because of its structure.

### 1.2.1 Traditional Banking System vs. Blockchain

In this subsection, we will point out some traditional banking system properties and how blockchain handles them. Some of them will be discussed in detail in later chapters.

When looking to a bank check, we can notice some important elements such as the signature of the sender. This element tells the bank that it was account owner who authorised the transaction. In blockain, this is handled with cryptographic signatures made out of hash functions discussed in chapter 2.

After an individual receives a check, he goes to the bank to withdraw the money. At this point, the cashier looks at the account's balance to verify if there is enough money to make the transaction. The account's balance is registered in several bank data bases. On the other hand, blockchain saves this information on its own structure element, a block, and replicates the information trough all the blockchain network. Then, it can be accessible to every party who owns a copy of the blockchain. This makes the blockchain a *distributed* ledger.

In terms of the transaction, if the sending account has enough money, then the charged amount will decrease in the balance of the sender and increase in the receiver's one. This will prevent the sender to spend money he already gave away and does not own anymore. Blockchain prevents this problem with the mining and Proof of Work protocol.

Because of its structure, the blockchain makes every historical transaction traceable and inmutable, thus the blockchain is a trusworthy registry.

### 1.2.2 Structure

The blockchain, as its name says, is made of blocks. Each block contains the information of several transactions, for bitcoin the size of the block is about 1MB. The block also contains a time stamp, the version of the blockchain, the merkle tree root hash, the previous block hash and a random value called nonce. This last one plays a fundamental role in the mining process, and is explained later on.

Once all this information is packed into a block, it gets encrypted with a hash function. Briefly explained, hash functions have the property that if having an encrypted message, it is extremely hard to reconstruct the original message. Also a hash function returns a totally different value from two messages that differ slightly. For example one message has a full stop at the end and the other does not. Having said that, we will explain how adding the previous block hash into the information of the current block makes it hard to modify the content of the transactions.

In order for a block to be valid, its hash needs to begin with a certain amounts of zeros. This will be better explained in the Mining subsection. But let's consider the following example. Let's imagine a 10 blocks length chain. If someone wants to modify a transaction that took place inside the third block then, the block hash will change radically and it won't begin with the required amount of zeros for it to be valid. But what will happen with blocks 4 to 10? Well, because the hash of block 3 is part of the information encrypted to generate block 4 hash, this change inside block 4 will also generate an invalid hash. Therefore, block 4 would be invalid. This will also invalidate the following blocks, then only blocks 1 and 2 will remain valid.

### 1.2.3 Mining and Proof of Work

In the past subsection, we mention that in order for a block to be valid, its hash must begin with a certain amount of zeros. Here is where the "nonce" plays an important role. Miners will get a different hash every time they encrypt the same block with different nonces. So mining actually represents the miners race using brute force to find the nonce that gives a valid hash so a new block gets added to the blockchain.

The amount of zeros will be set dynamically in order to mine a block every certain time. In bitcoin, a block is added to a blockchain every 10 minutes. So if there are a lot of miners the difficulty is increased by decreasing the number of zeros. In bitcoin, the first miner to add a new block to a blockchain gets paid, so miners have the incentive of spending more computing resources in order to solve the puzzle first. This is called Proof of Work (PoW).

There are some others protocols designed to reach consensus even when there are byzantine faults, such as Proof of Stake or the PBFT used for Permissioned blockchains. We will also be talking about this later.

### 1.2.4 Properties

Throughout these pages, we will point out some of the most important properties of blockchain, such as it as being secure, traceable, tamper-proof and so on. Also we will show some alternative designs for blockchain and some hazards present while writing in solidity

### 1.2.5 Applications

Beyond Cryptocurrencies, we will be talking about where can we apply this technology. We will set special attention for 3 applications: Supply Chain using IOT, IOU Credit Networks and Atomic Cross-Chain Swaps. These and other applications will require to scale the solutions to bigger problems in terms of transactions, or storage, so we will also present a scalability solution via Layer 2.

### 1.2.6 Regulation

As many technologies, blockchain has its own regulatory issues. In chapter 9.2.1 we will talk about some privacy challenges, solutions and unresolved issues of this technology. Another regulatory issue discussed is the custody that takes place within the wallets and the keys they must keep in a secure way.

### 1.2.7 Other Issues

Last but not least, we will present a few pages of the state of art of this controversial technology. Despite of being announced as a secure technology in which only by controlling 51% of the network can achieve a successful attack , we will present an attack in which only 33% is necessary. This and other unresolved issues represent challenges for those working on blockchain development and research. Take this work as an introduction to the amazing world of blobkchain, understanding that there is still much to be done.

# Chapter 2

# Cryptography and Security

Talk by: Amir Herzberg
Chapter by: Fernando Arreola and Erick Chávez

A blockchain, as the name indicates, represents a chain of blocks, but what kind of chain? and, of what kind of blocks? In the following sections, we will introduce some concepts of cryptography that make possible this technology. First, we will make a brief review of hash functions and some important desirable properties of these functions. Second, we will analyse the hash blocks. Third, we will present a blockchain itself.

## 2.1 Hash functions

A hash function $h$ is a function that maps an input $m$ of arbitrary length to a fixed size output $h(m)$. In general, we want the output to be shorter than the input (for this reason, the output is normally called *digest*). Hash functions have many applications like cryptography, security and efficiency. It is easy to see why hash functions deliver efficiency, without introducing other properties, we could say that it is more efficient to store the shorter strings, generated by a hash function, than the original strings in memory for example.

Given these definition, we can think of many examples of hash functions. A simple one is the hash function $h_{LSD}$ that given an input $m$, a natural number in base ten, returns $h_{LSD}(m)$, the least significant digit of this input. For example, for the hash function $h_{LSD}$, if $m = 17$, then $h_{LSD}(m) = 7$; if $m' = 133$, then $h_{LSD}(m') = 3$; and if $m'' = 127$, then $h_{LSD}(m'') = 7$.

Another simple example is the hash function $h_{SUM}(m) = (m_1 + m_2 + ... + m_n)$ mod 10 which maps an integer number (domain) to the sum of its n digits (range) modulo 10. For example, for the function $h_{SUM}$; if $m = 17$, then $h_{SUM}(m) = 8$; if $m' = 133$, then $h_{SUM}(m') = 7$; and if $m'' = 127$, then $h_{SUM}(m'') = 0$.

These hash functions are easy to compute and they also map a string of an arbitrary length to a fixed length output. But, we also want other properties in hash functions that aim to protect money or other valuable assets.

### 2.1.1 Hash functions properties

A very important property of hash functions, related to efficiency, is that when we apply a hash function to a set of values, the *hashes* should be distributed uniformly in the range. Also, this mapping should behave like a random permutation. In other words, if we have a set of inputs, it should be mapped randomly into equally-sized bins. These hashes are *pseudo-random*.

A hash function is said to be *collision resistant* if it is hard to *find* two values $m \neq m'$ such that $h(m) = h(m')$. Since $|domain| >> |range|$, collisions are possible, but they should be *hard* to find. Consider the functions introduced in the previous section, it is not a hard task to create a collision between functions $h_{LSD}$ and $h_{SUM}$. Given a hash, an adversary can easily find infinitely many inputs that map to the same output. If it is hard to find collisions and the adversary tries at random, she should try up to $2^n$ different input values (all different combinations of 1 and 0 in a string of length $n$). In figure 2.1 we see a collision between two messages, in this case, the function $h$ maps a string of bits of arbitrary length, $\{0,1\}^*$, to a string of bits of length $n$, $\{0,1\}^n$.

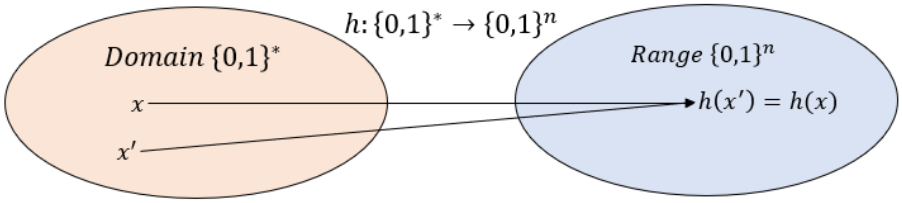


Figure 2.1: Hashes Collision

An important application for collision resistant functions is checking integrity of an object/file/message. The motivation is as follows, suppose you have the hash of some secret (which is not the actual secret) and the function to hash the messages, then someone says to you that he has the secret, how do you know if he is telling the truth? If he gives you the message and it produces the same hash, then he must certainly be telling the truth. Another approach would be to consider a simple example where someone can download a software directly from the developer $m$ (secure) and some other user, far away from the developer data centre, may download the software from a repository $m'$. How does a

user know if the software in the repository and the data centre are the same ($m = m'$)? We can use a hash function to hash the bits in the software provided by the developer producing $h(m)$ and making this available to everyone. After comparing the hashes, if $h(m) = h(m')$, the software is $m = m'$ with probability $1 - 1/2^n$ (assuming independence between bits). As you can see, it is efficient to verify a message. In the next subsections we will explore more deeply these properties in order to understand why they are useful in blockchains.

## 2.2 One Way Functions

Another essential and very important property of hash functions is pre-image resistance. We define this property as follows: Given $h(x)$ for random $x$, it is hard to find $x$ or any $x'$ such that $h(x') = h(x)$. In other words, given an input $x$, if a hash function $h(x)$ produces a hash value, let's say $z$, then it should be a difficult process to find any input value $x'$ that hashes to $z$. In figure 2.2 we see a graphical representation of this:

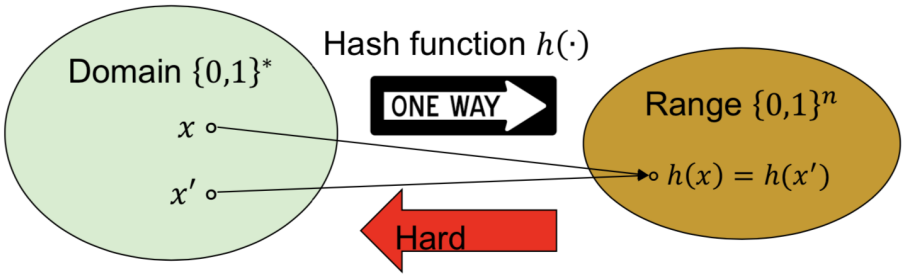


Figure 2.2: Pre-image resistance

This property means that it should be computationally hard to reverse a hash function, which implies protection against an attacker who only has a hash value and is trying to find the input. One application of one way functions is in *one time password authentication*, method that generates a password that is valid for only one login session or transaction. We can extend this method to one-time signature, that works as follows for a single message $b$:

- Private key: Two random one time passwords, $s_0$ and $s_1$
- Public key: Two strings, that consists on the hashes of the one time passwords, $v_0 = h(s_0)$ and $v_1 = h(s_1)$
- Signature: To sign $b$, send signature $\sigma = S_{(s_0,s_1)}(b) = s_b$
- Validate: $V_{(v_0,v_1)}(b, \sigma) = \{1 \; if \; v_b = h(\sigma), 0 \; else\}$

For example, Alice wants to send a message to Bob. Initially, she has to establish a random private key. Second, she has to create a public key, making use of a hash function. Next, using a hash function and her private key, produces a sequence of random values and signs the message. Finally, Bob receives the message, and verifies the integrity of it, using Alice's public key to generate a sign and compares with the sign that receives with the message. If both are the same, the message is valid.

### 2.2.1 Difficulty to find $x$

We mentioned that is hard to find $x$ or any $x'$ such that $h(x) = h(x')$, but how hard? Intuitively, must do about $2^n$ hash computations, where $n$ is the length of output of hash, tipically $n > 80$, that is not computationally feasible. Even if we make smaller the target output set, called $Y$ with size $m$, to find an input $x$ such that $h(x) \in Y$, we must do about $2^m$ hash computations (as we can see in 2.3), that is less tan $2^n$, but it is improbable to find hashes easy to invert. We will see, in next chapter that in order to incentive *miners* to add new blocks to the blockchain we may decrease the difficulty by making decreasing the length $m$ of the string that the *miners* must found. This concept is known as *proof of work*. So, proof of work is a method in which miners know that it in order to add a new block to the blockchain, and obtain some prize, they will need to spend a large amount of computational resources, but it is not impossible, since we do not need to get the complete hash, just a substring. This is increased regularly since computational capabilities also increase periodically, this way, the relative difficulty is maintained.

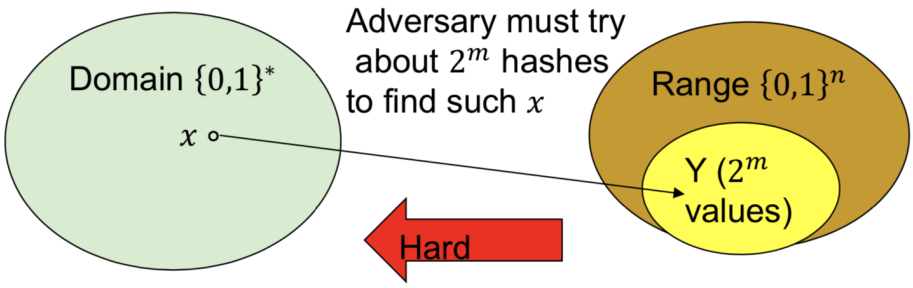


Figure 2.3: Subset Y

### 2.2.2 Second preimage resistance.

We define this property as follows: Hard to find collision with random $x$, for example, $x'$, such that $h(x') = h(x), x \neq x'$. In other words, if a hash function $h$ for an input value $x$ produces hash value $h(x)$, then it should be difficult to find any other input value, lets say $y$, such that $h(y) = h(x)$.

This property protects against an attacker who has an input value with the corresponding hash, and wants to substitute the original input value. In figure 2.2 and 2.4 we can see a graphical representation of this property. In this way, this type of hashing is different from *locality sensitive hashing* (LSH) which wants to map similar items to the same bucket.

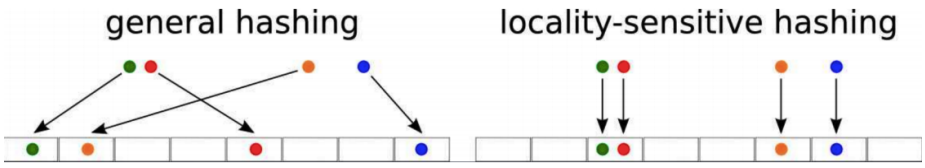


Figure 2.4: Crypto hash vs LSH

Now that we understand the properties of cryptographic hash functions, we will see how to combine hashes to create a blockchain.

## 2.3 Hash Block

Now that we understand hash functions, we will introduce the concept of hash blocks. It is important to remember that we can hash any string of arbitrary length to a string of fixed length. So we can think, why not combine the hash of many functions into one hash. This way, we are able to hash many secrets into a simple hash and we can verify many messages with only one hash, this way, we can be more efficient and require less communication. Consider the structure in figure 2.5, where we can input hashes of previous messages into a hash function in order to get a hash that considers *all* the previous hashes, this way, we can verify the information of $m$ messages with just on hash.

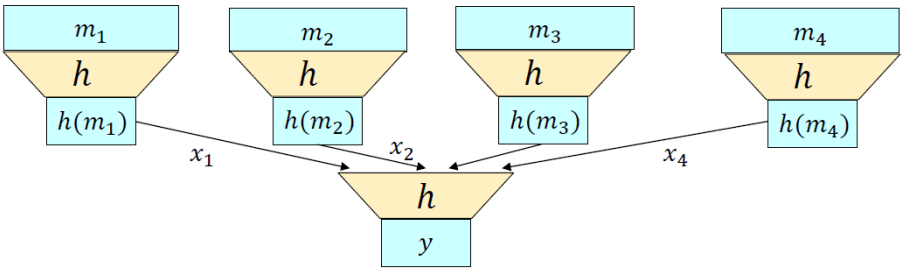


Figure 2.5: Hash block

Now, we will follow the example presented in figure 2.6. Consider messages $m_1, m_2, ..., m_n$ now, we use the hash function $h$ to hash this messages, producing $x_1 = h(m_1), x_2 = h(m_2), ..., x_n = h(m_n)$ and then, we hash all this hashes to produce $h(x_1||x_2||...||x_n)$, where $||$ is a concatenation operator.

In this case, suppose we want to verify if a message $m' = m_2$ and we have the hashes $x_1, x_3, ..., x_n$, so we can hash $m'$ to produce $x' = h(m')$, then if $h(x_1||x'||...||x_n) = h(x_1||x_2||...||x_n)$ it is almost certain that $m' = m_2$. As we saw, hashes can be *nested* in order to maintain efficiency, hashing the concatenation of hashes is just a hash, so the original properties are maintained.

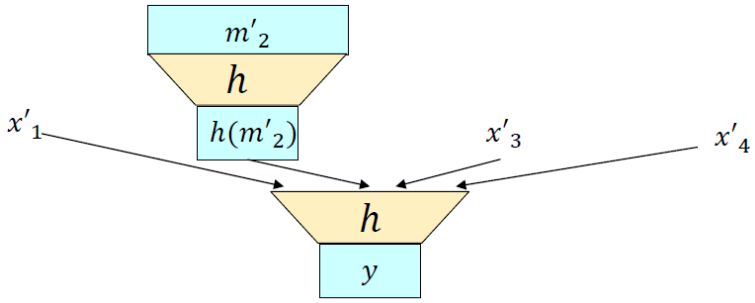


Figure 2.6: Hash block: validating message $m' = m2$

We must be thinking, so the *blockchain* may be just a concatenation like the *hash* and the *hash block*. That is true, we will use another level of hashing in order to hash blocks into just one hash.

## 2.4 Blockchain

As we said earlier, we can *nest* hash structures in order to maintain efficiency on increasing levels of complexity. Consider the blockchain structure in figure 2.7. This hash structure is created as follows: in the beginning, we have messages $m_1^1$ to $m_1^n$, where the sub-index is the message index and the super-index is the block index, these messages are hashed individually and then these hashes are hashed into a *hash block* $h(B_1)$ and, finally, this hash block is hashed with a hash function $h$ into a link $\delta_1$ of the *blockchain*. Remember that this hash is of a fixed length.

Suppose we need to add a new block $B_2$, we follow the same process to make the hash block and, we will use $h$ with inputs $\delta_1$ and $h(B_2)$ to produce $\delta_2$. In general, if we have $\delta_t - 1$, in order to get $\delta_t$, we will use the hash block $h(B_t)$ and $\delta_t - 1$ to produce $\delta_t$. So, as we mentioned, the computation only requires the previous digest and the current block.

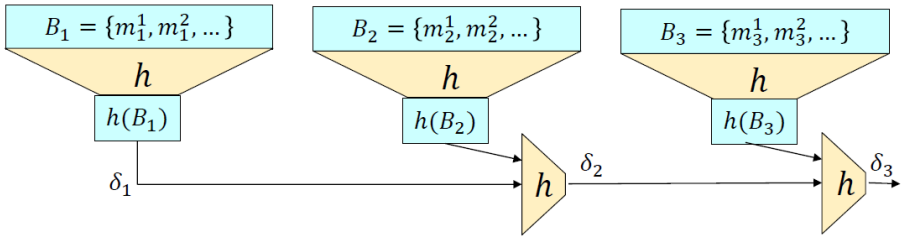


Figure 2.7: Blockchain

As with the previous hash structures, is important to understand the validation of any item. Suppose we are currently on $\delta_3$, and we want to validate a message $m_2^2$ of a previous block $B_2$. What is the process we need to follow to validate this message? First we need to receive all the information, namely $\delta_1, h(B_3), m_2^2, x_1, x_2, x_3, ....$ We can follow the process in figure 2.8. So we will know if our message is valid, $m_2^2 = x_2$, if after calculating the hash of $h(B_2)$ and producing $\delta_2$ this $\delta_2$ with $h(B_3) = y_3$ produces $\delta_3$.

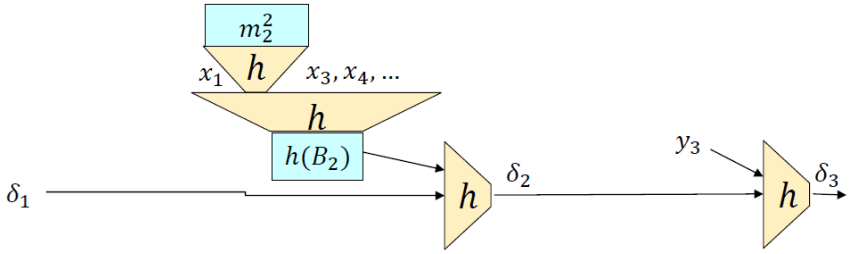


Figure 2.8: Blockchain: validating a message

As you can see, with little information, we can validate, any component of the blockchain. We do not need to send all the information, just the hashes. Therefore, this structures are very efficient. Also, we never shared the actual information, there was no exposure, just the hashes, and since it is difficult to know what message produces a *specific* hash, it is difficult to tamper the information.

In the next chapters we will review some considerations that wee need to have in mind when we want to implement a blockchain like: types of blockchains (public or private), specific domain programming languages for blockchains, smart contracts and applications.

# Chapter 3

# Permissioned Blockchains

Talk by: Amr El Abbadi
Chapter by: Gabriela Vargas and Fernando Arreola

Blockchain applications may vary between different domains such as finance, supply chain, healthcare, insurance, and real state, to name a few. The emergence of public blockchain ledgers like Bitcoin and Ethereum has brought many changes to the world of finance. In previous years, transactions in traditional financial systems involved the active participation of third parties who validated transactions, but blockchain has now enabled individuals to complete transactions without needing the participation of intermediaries.

Aside from the cryptocurrency world, there are other types of problems that can be solved with the implementation of distributed ledgers. These ledgers do not need to be open to everyone since confidential information might be shared across private environments, in which participants actually know each other, but do not trust each other.

Consider a supply chain environment, compounded by different entities such as manufacturers, materials suppliers, transport suppliers, warehouse suppliers, distribution centers, wholesale buyers and individual buyers. In this scenario, all the actors have a common goal, which is delivering the final product to the consumer in proper quality conditions and in a certain amount of time, but the problem is that these entities do not trust each other, as each of them might have different incentives.

The mistrust behind the above-mentioned scenario is the reason why blockchain can be presented as a suitable solution to improve product traceability through the stages of the supply chain. Rather than being an environment where everyone can join and participate, this situation demands a private solution that enables consensus between authorized agents. These ledgers are known as Permissioned Blockchains and the most important capability sought in these systems is transparency so any authorized party can access the information and audit transactions.

Given that permissioned blockchains run along with a set of known, identified participants, traditional consensus protocols can be used to add transactions to the ledger. This means that they do not longer need the participation of miners nor the implementation of consensus protocols that are used in public ledgers. It is important to remember that both public and private blockchains seek to achieve similar characteristics in records such as immutability, transparency, provenance, fault-tolerance, and authenticity; but in the case of private blockchains, there are significantly fewer agents involved in transactions, so less intensive consensus protocols can be implemented to secure interactions.

We provide a summary of the differences between public and permissioned blockchain networks in Table 3.1

| **Characteristic** | **Permissionless** | **Permissioned** |
|---|---|---|
| Participants | Anonymous, could be malicious | Known, Identified |
| Consensus Mechanisms | Proof of Work, Proof of Stake<br>Large energy consumption<br>No finality<br>51% attack | Byzantine fault tolerance (PBFT)<br>Lighter<br>Faster<br>Low energy consumption<br>Enable finality |
| Transaction Approval Time | Long (Bitcoin: 10 min or more) | Short (100 x mseg) |

Table 3.1: Permissionless vs Permissioned Blockchain

## 3.1 Practical Byzantine Fault Tolerance (PBFT)

Practical Byzantine Fault Tolerance (PBFT) is the consensus protocol commonly adopted by enterprise consortiums in trust-less environments. Its goal is to implement a deterministic replication service with **arbitrary malicious faults** in an asynchronous environment.

Byzantine fault tolerance (BFT) is the property of a system capable of withstanding failures derived from the Byzantine Generals Problem. The problem consists in a group of generals, each one with their own army, located in different places around the city they are trying to attack. The generals must agree to attack or withdraw. It does not matter if they attack or withdraw, as long as all the generals reach a consensus, that is, they agree on a common decision to execute in coordination. As the generals are located in different spots, communication failures may occur, which means that messages could be delayed, destroyed or lost on their way. Moreover, even if a message is successfully delivered, one or more generals may choose to act maliciously and send a fraudulent message to confuse the other generals.

In the blokchain context, each general is represented by a node and the nodes must reach a consensus about the current state of the system. In other words, most participants within a distributed network have to agree and execute the

same action to avoid a system failure.

PBFT provides safety and liveness to the Byzantine Generals Problem assuming at most $(n-1)/3$ faulty nodes out of $n$ total nodes. This implies that clients eventually receive response from servers and this response will be correct [2]. PBFT consists of three main phases:

1. *Pre-prepare*: Picks order of requests
2. *Prepare*: Ensures order within views
3. *Commit*: Ensures order across views

The number of nodes integrating the network is critical, as it will determine the message count, due to amount of multicast messages sent on each stage of the protocol.

## 3.2 Permissioned Blockchain Systems

In this section, we will describe three examples of permissioned blockchain systems: Tendermint, Hyperledger Fabric and ParBlockchain.

### 3.2.1 Tendermint

Tendermint establishes decentralized trust in a network of known nodes using a Byzantine Fault-Tolerant Protocol which considers a specific node type called *validator* that has certain voting power according to the amount of coins it has locked in a *bond deposit*, similar to the Proof of Stake Protocol used in public blockchains.

In Tendermint, a block is said to be valid if all of its transactions are valid and sufficient validator signatures are included in the validation [3]. Consider a simple transaction: "Transfer $x$\$ from account $A$ to account $B$. The transaction will be valid if (1) The owner of $A$ initiates the transaction, and
(2) $A.balance \geq x$. Likewise, a block is valid if (1) All the transactions in the block are valid and (2) Sufficient signatures (2/3 majority of validators) are included in the validation.

The consensus protocol consists of the following steps:

1. Each round has a designated proposer (leader) chosen in a round-robin fashion.
2. Blocks are committed in a chain, with one block at each height.
3. A block may fail to be committed, in which case the protocol moves to the next round, and a new validator proposes a block for that height.
4. Stages of voting: pre-vote and pre-commit (Prepare and commit in PBFT)
5. A block is committed when more than 2/3 of validators pre-commit for the same block in the same round.

When a validator signs duplicitously, an evidence transaction can be generated by anyone with the two conflicting commit-vote signatures. This evidence is commited into the blockchain and the guilty validator's bonded coins will be destroyed. Similarly to PBFT, Tendermint assumes that less than 1/3 of the validators are malicious, otherwise, a 2/3 majority of validators would sign two blocks at the same time and a fork will be generated.

Tendermint's main issues are about performance, as sequential ordering and execution of every transaction needs to be performed by every validator node; and confidentiality, as every node accesses the data on each transaction.

### 3.2.2 Hyperledger Fabric

This system relies on smart contracts to execute transactions. Smart contracts are computer programs that self-execute once they are established and deployed. The logic of the contract is triggered to be executed once some conditions or terms are met. In contrast with other public and permissioned blockchain models, Hyperledger Fabric supports the execution of distributed applications written in standard programming languages [4], instead of domain specific languages such as Solidity.

Following an Execute-Order Validate architecture, transactions in Hyperledger Fabric are first executed, then ordered, and finally, validated. This model consists of the following steps:

1. Each transaction (of an application) is first executed by a subset of nodes (**endorsers** of the application).

2. A separate set of nodes (**orderers**) orders the transactions, puts them into blocks, and multicasts them to all the nodes.

3. Each node validates the transactions within a block and updates the ledger

Hyperledger Fabric presents performance issues in workloads with contention, when two transactions access the same record and one of them is a write operation.

### 3.2.3 ParBlockchain

This blockchain was proposed to address contention problems presented in other permissioned networks. It is designed under the OXII paradigm, which supports workloads with different levels of contention. This paradigm follows an Order-Execute model, meaning that transactions are first ordered, and then executed; it also supports parallel execution of non-conflicting transactions in the same or different applications.

OXII considers two types of nodes: orderers and agents. Orderers determine the order of the transactions of different applications, build the block of the transaction and then generate a *dependency graph* of the transactions within the

block. Next, a group of agents of each application called **executors**, perform the transactions of their application and multicast the results [5].

As we can deduce from the description of OXII, conflict detection is enabled in these type of systems, as any contention between transactions is detected in the ordering phase and considered in the execution phase.

In case of two conflicting transactions executed in parallel, the result must be identified as invalid. This kind of inconsistent results can be detected in the last phase with its consequent negative effect in performance.

# Chapter 4

# Alternative Design for Blockchains

Talk by: Amr El Abbadi
Chapter by: Dalia Camacho and Gabriela Vargas

Public blockchains have serious drawbacks in terms of efficiency. They have a very high latency for transaction confirmation, due to the waiting time necessary to confirm that the transaction belongs to the longest branch in case a fork occurs. This requires in average that other six blocks are added to the blockchain. Since adding a new block on the chain takes about ten minutes, the confirmation takes about an hour. Also, throughput is very low, because there are only three to seven transactions per second (TPS) compared to the $24,000$ TPS that Visa manages.[6]

The most straight forward solutions would be to increase block size or decreasing the time of adding a new block. However, some problems arise from these approaches.

By increasing the block size more transactions could be added in a single block, however, the bandwidth of the network and storage should also increase. As externalities of building larger blocks verification time would increase and fairness between miners could disappear.

To decreasing the time of adding a new block, we would need to decrease the difficulty of mining. However, from the proof of work (PoW) approach this would be counterproductive. If it was cheap to add new blocks the blockchain would be prone to Sybil attacks. Another issue would be an increase in the number of forks, due to an increase in the probability of two miners being able to solve the problem.

Ideally we would like to have a byzantine fault tolerant protocol such as Practical Byzantine Fault Tolerance protocol which allows a third of the processes to fail. However, such protocols require all participants to communicate with each other, which is not scalable for public blockchains.

In order to solve these problems new alternatives for adding blocks have been proposed such as:

- **Bitcoin NG:** Mine once, publish transactions many times.
- **ByzCoin:** Form a committee to vouch for new block.
- **Elastico:** Shard transactions across different committees

These approaches will be described in the following sections.

## 4.1 Bitcoin NG

The main idea of bitcoin NG is that a miner who solves the PoW will be elected as a leader and he can add serialized transactions into the blockchain until another miner solves the PoW. [7]

Blocks created by PoW are called *key blocks*, they contain the public key of the miner, the nonce of the PoW and are meant for leader election. Blocks added serially by the leader are called *micro blocks*, these blocks contain the transaction and are signed with the leaders private key. Both key blocks and micro blocks contain a reference to the previous block.

The Bitcoin NG approach solves the issue of throughput, because adding new transactions into the blockchain is an easy task for the leader.

Miners are remunerated for their work in two manners, the first is a fixed amount when adding a key block to the chain. The second is a percentage from the transaction fees in every transaction added to the block. Here the current leader earns 40% of the transaction fees and the next leader will earn 60%. Miners can only spend their earnings after 100 key blocks are added. [7] This is an incentive for miners to follow the chain instead of adding a new fork.

However as a leader one could intentionally create a fork by adding different microblocks to the different branches which may cause double spending. To avoid this a special transaction called *poison transaction* can be placed indicating the branch in which the malicious fork appeared. The latter is called a *proof of fraud* and invalidates the compensations to the leader that caused the fork.

Even though the Bitcoin NG alternative has a better latency and throughput than the current approach it still has some issues. It is not consistent due to the occurrence of forks whenever a new key block is added, because the previous leader will continue mining. Also, selfish mining can still occur.

## 4.2 ByzCoin

The aim of ByzCoin approach is to enhance Bitcoin security and consistency by a PBFT protocol and collective signing.[8] For PBFT to take place a closed group is required, which by definition does not occur in a public blockchain. However, the PBFT can take place between a subset of the nodes which conforms a consensus group.

There could be several ways to select the consensus group, but some, such as open membership, are prone to Sybil attacks. Thus, PoW is done to define membership. Blocks added by PoW are also referred to as *key blocks*.

The consensus group has a fixed size, once a miner solves PoW to *proof membership*, this miner is given a *consensus group share* and the oldest share expires. Therefore, we can see this process as a *sliding share window*. The new miner will become the leader that will propose the transactions for new blocks. The voting power will depend on the amount of key blocks a miner has within the current window.

The remuneration given to the miners will come from transaction fees which will be divided between all miners in the consensus group.

The blocks that contain transactions are called *microblocks* and those created by PoW are called *key blocks*. There are two separate chains for each block type to avoid race condition.

Micro blocks are decided by the consensus group using the PBFT protocol. Every microblock must be validated and signed by two thirds of the trustees in order to add the transaction to the chain. To achieve scalability they combine the CoSi protocol for collective signing with Schnorr multisignatures.[8] There are four steps within the CoSi protocol.

1. **Announcement:** equivalent to pre-prepare in PBFT
2. **Commitment:** equivalent to prepare in PBFT
3. **Challenge:** equivalent to proof-of-acceptance in PBFT.
4. **Response:** equivalent to commit phase in PBFT.

This approach has two main benefits. The first is that double spending by a malicious leader is avoided since it can be verified with the CoSi protocol. The second one is that forks in the microblock chain are non existent, therefore the validity of a transaction is immediate once it is on the microblock blockchain and there is consistency in all the nodes. Forks in the key block blockchain may occur, but the winning block is decided in a deterministic manner.

## 4.3 Elastico

The idea behind Elastico is to parallelize the execution of adding new transactions into the blockchain. Miners are divided into different committees and each committee processes an independent set of transactions or *shards*. A final committee called the *consensus committee* combines all the shards processed by the other committees and broadcasts the transactions into the blockchain.[9]

There are five steps within the Elastico protocol.

1. **Identity Establishment and Committee Formation:** Each miner solves PoW and combines the latter with an IP address and a public key to generate an ID for a given epoch. Each miner is assigned to a committee

depending on their ID. If there are $2^s$ committees then the last $s$ bits of the ID define the committee to which the miner belongs to.

2. **Overlay Setup for Committees:** In every epoch the IDs change, therefore to identify other committee members there are two main options. The naive one consists on broadcasting the committee, but this is $O(n^2)$. The other option is having a specialised directories for each committee. With the latter approach identifying other members takes $O(c\dot{n})$ where $c$ is the number of committees.

3. **Intra-committee Consensus:** Each committee runs a classical byzantine agreement protocol such as PBFT to select a set of transactions. Once the shard has been selected it is sent to the *consensus committee*.

4. **Final Consensus Broadcast:** The final committee runs a classical byzantine agreement protocol from the different shards to obtain the block that will be added to the blockchain.

5. **Epoch Randomness Generation:** A set of random values is generated and broadcasted through the network for the next PoW problem.

This solution prevents fork generation and consistency is guaranteed.

# Chapter 5

# Solidity Pitfalls and Hazards

Talk by: Maurice Herlihy

Chapter by: Dalia Camacho and Bernardo García Bulle

Solidity is a high level language for the Ethereum Virtual Machine (EVM) to create smart contracts. It is based on JavaScript, which is a wide spread programming language and is in general easy to use.

However, choosing JavaScript as a basis for Solidity may not have been the best idea. JavaScript is meant for webscripting, while the aim of smart contracts is to execute irrevocable financial transactions. Not only is the aim quite different, but there are some odd and complex behaviours within JavaScript that are taken for granted. Moreover, there is no precise notion of correctness within JavaScript. Thus, smart contracts in Solidity are prone to errors that have been catastrophic. However, currently no other (or better) options exist.

Here we review some of the previous problems such as the re-entrancy attack, the silent-overflow attack, the unexpected ether attack, and the delegatecall attack.

## 5.1 Re-Entrancy Attack

Solidity allows for high modularity, thus contracts can call one another. In the re-entrancy attack a contract calls an external contract that unexpectedly calls the first contract again. This attack allows transfering ether from someone else's account several times until the account runs out of ether.

As an example on how the re-entrancy attack may work consider the following functions.

```
1 function withdraw(uint amount) {
```

```
2       client = msg.sender;
3       if (balance[client] >= amount} {
4           if (client.call.sendMoney(amount)) {
5               balance[client] -= amount;
6       }}}
```

```
1   function sendMoney(uint amount) {
2   balance += amount
3   msg.sender.call.withdraw(amount)
4   ...
5   }
```

The function withdraw first verifies a client has at least some amount. If this is true then the function sendMoney is called and after that the corresponding amount is subtracted from the client's account. The problem occurs when calling the function sendMoney. this function increases the balance in a given amount, but instead of finishing there it calls the function withdraw once again. At this moment the amount has not been deducted from the client's balance, but the client can withdraw once again. This can be repeated indefinetly.

The Descentralized Autonomous Organization (DAO) was attacked by the re-entrancy attack in June 17 2016 who lost 3.6 million Ether.[10] To solve the problem a hard fork was done and all transactions were rolled back, so the DAO did not lose the money. This hard fork went against the idea of the Ethereum blockchain being a descentralized untamperable ledger.

Re-entrancy attacks can be prevented by changing the state before entering another contract, by locking external calls when a contract is already working, or by allowing very little gas[1] to be spent in the transaction.

## 5.2 Silent-Overflow Attack

The second problem we analyze is the silent-overflow attack. In solidity integers are of a fixed size, and values rollover when an overflow or an underflow occurs, and there is no warning or exception error when this occurs. For example.

```
1   uint8 x = 0;
2   uint8 y = x - 1; // y is 255
```

A problem that may occur when verifying if a client has enough money on their account. The most obvious way to do this is subtracting the amount to the current balance of the client and verifying the result is greater than or equal to zero. If a value larger than that of the client's balance is subtracted, the value will underflow and as a result the subtraction will always be greater than or equal to zero. Thus a true value is always returned. This is exemplified in the following program.

```
1   pragma solidity ^0.5.2;
2   contract Token {
3   ...
```

[1] Gas is an amount of ether that is payed to execute a Solidity program. If the execution exceeds the amount of gas established, then the execution stops.

```
function transfer(address _to, uint _value)
    public returns (bool) {
    require(balances[msg.sender] - _value >= 0);
    balances[msg.sender] -= _value;
    balances[_to] += _value;
    return true;
}
```

To prevent underflow and overflow issues it is best to use the SafeMath library instead of using arithmetic operations directly.

In real life attackers used the underflow vulnerability to their own advantage. They used the Proof of Weak Hands application on Ethereum, which offered PoWH tokens that increased in value when bought and decreased in value when sold. This application worked like a pyramid scheme or a Ponzi scheme. The attackers used the underflow to obtain tokens they did not have, then they sold them and were able to extract $800,000 from Ethereum. [11]

## 5.3 Unexpected Ether Attack

The third hazard is the unexpected ether attack, which occurs when someone else sends you ether. We can be careful when writing code to avoid someone stealing from our account, but we cannot always avoid receiving ether. Receiving ether without expecting it to happen could cause some problems we will see later on.

We must consider the ways in which we can receive ether. These are receiving ether from a named function, from a fallback function, or from a contract that self destructs an account. When receiving ether from a named function or from a fallback function we can revert this action by throwing an exception or by not being payable. But if a contract self destructs an account and sends all of its current ether to another account, this cannot be reverted.

As an example the problem of receiving unexpected ether may arise from a simple gambling game. Assume every player sends a predefined amount of ether to the main account, and when a milestone is reached the player who reached it will receive all the money.

Consider the following contract, where the amount players can send is exactly 0.5 ether, then the milestone should be a multiple of 0.5. If an account with 0.00001 ether selfdestructs and sends that amount to the account where the game is taking place, then the milestone will never coincide with the balance in the account. Since milestones are verified in terms of equality the conditions will never be met and ether in the account will be frozen forever.

```
function play() public payable {
    require(msg.value == 0.5 ether);
    uint bal = this.balance + msg.value;
    if (bal == mileStone1) {
        ...
    }
    else if (bal == mileStone2) {
        ...
```

```
9       }
10      else if (bal == mileStone3) {
11          ...
12      }
13      return;
14  }
```

To prevent this kind of attack we should avoid equality comparisons and we should have a payable function that tracks all transfers.

## 5.4 DelegateCall Attack

The last attack we explain here is the DelegateCall Attack. There are two ways in which functions from other contracts can be called. The first one, `call`, calls the function remotely. While, *delegatecall*, makes a copy of the function within the caller's contract and executes it in this context. Let's assume contract A does a delegatecall of contract C, then A has a copy of C. However, if contract C is selfdestructed, then when contract A does a delegatecall to C, contract A will also selfdestruct.

This is what occurred with multisig wallets. Every private wallet made a delegate call to a wallet library. This library had no owner, then an attacker took ownership of the library and initialized it. Afterwards this attacker selfdestroyed the wallet library. Having done this all the wallets that used the multisig wallet library selfdestructed when calling it. All the ether from these wallets which was about $300 million was lost forever. In this case no hardfork was made.

To avoid these kind of attacks we should always initialize a contract and define ownership to prevent an attacker from taking ownership. Other solutions would be to eliminate the selfdestruct function and to define a contract as stateless when it is defined as a library so no one can become an owner.

# Chapter 6

# Blockchain for Supply Chain and IoT

Talk by: Aniket Kate

Chapter by: Edgar Granados and Carlos Valenzuela

## 6.1 Blockchain technology

### 6.1.1 Core industrial Architecture

The blockchain architecture can be divided into four: blockchain, replications, digital signatures (PKI) and a consensus mechanism.

The *blockchain* is a database whose primary function is to aggregate transactions in blocks in such a way that each block can be added to a chain of already existing blocks. To share the blockchain across an existing network *replications* are used. The replication allows all participants to have their own *identical* copy of the ledger. In order to control the permissions that each participant has within the ledger, *digital signatures* are essential (usually implemented as a Public Key Infrastructure). Finally, it is essential to have a mechanism that ensures that each asset can only be used once. *Smart contracts* are the standard approach although this is also known as a *consensus mechanism*.

### 6.1.2 Current Applications

Traditionally, blockchain has been used in financial systems in the form of crypto-currencies such as *Bitcon* and *Ethereum*. It has also been implemented as payment settlement systems such as *Ripple* and *Stellar*. Finally, what has been called the *Internet of Value* is in development.

However, blockchain also has the potential of being implemented outside of the financial system. Supply chain and identity Management are two areas that

can benefit significantly from a blockchain implementation.

## 6.2 Traditional Supply Chains

A supply chain refers to the upstream and downstream relationship with suppliers and customers to deliver superior customer value at less cost to the supply chain as a whole [12]. The supply chain of a specific company can be viewed as a directed graph where each supplier is either connected to another supplier or has a direct connection to the company. The company's node has a connection to each customer (each of which may have a connection to other customers).

For a company, an integrated supply chain is a tool that seeks to improve quality and service while lowering costs, providing a competitive advantage. It is important to note that integrated supply chain exchanges not only goods but also knowledge. By exchanging company knowledge as part of the supply chain, the company can improve operational performance as it can successfully establish optimal operations within his suppliers and customers. In this way, an integrated supply chain can be viewed as a network of connected and interdependent organizations mutually and cooperatively working together to control manage and improve the flow of materials and information from suppliers to end users [13].

As supply chains represent a vital asset to organisations, they are in constant improvement. It is not difficult to see why blockchain represents a good alternative for *traditional* supply chains; however, the introduction of blockchain to *traditional* represents essential challenges. Some of these challenges include identifying counterfeit, theft of intellectual property and fault detection.

## 6.3 Blockchain for Automotive Supply chains

In [14] the authors present a system using blockchain to identify and prevent counterfeiting. By using RFID, the system manages to track the products along the supply chain. By using Blockchain conflicting transactions can be detected, and once a transaction is added it cannot be removed. The solution follows three steps: first, it creates a smart contract where all participants agree on a set of rules and logic. Then the smart contract is deployed: the agreed rules and logic are installed in a set of *validators*, which initialize the blockchain. Finally, each creation or transfer of products is added as transactions following the smart contracts.

The implementation is composed of a *Hyperledger* architecture (which implements a *Execute-Order-Validate* logic instead of the traditional *Order-Execute*) using *Golang* smart contracts achieving a flexible and effective supply chain management. However, there are key issues for this type of solutions: scalability, intellectual property protection, and meta-data privacy. The IP-protection problem is best to exemplify in a recent struggle among different cargo companies, where Maersk (the biggest cargo company in the world) partnered with

IBM to offer a blockchain solution. For their solution to work, it needs that all or at least most of the cargo companies join the network. However, the companies are worried about what it means that the major cargo player is part and owns the network.

## 6.4 Blockchain for AI Marketplace

Another challenge for blockchain is to create a decentralized Artificial Intelligence marketplace. The main problem that some AI applications are currently facing is data ownership and security. In [15] a review and open challenges of the integration of blockchain for AI are presented. By integrating blockchain in AI applications, numerous benefits can be achieved such as enhanced data security, improved trust, collective decision making, decentralized intelligence, and high efficiency. Apart from giving security and privacy guarantees for data owners, blockchain could also give companies and startups IP security.

## 6.5 Blockchain for Digital Supply chains

Traditional *Digital supply chains* usually work by developing software and sharing it with authorised users (i.e. licence buyers for software utilisation). However, it is almost impossible to stop the copying/ replication of the data once it is shared with them. Matters get worse once we consider that software modification without changing its functionality is easy, while imputing blame on possible malicious intermediaries disclosing or modifying the software is difficult. Even though legal systems recognise unauthorised copy, distribution or modification of software as a felony, as it violates terms of licence, it is cumbersome and costly for companies to prosecute contract violators.

For that matter, Aniket Kate and its collaborators developed a software that introduces *Crypto augmented Blockchain Smart Contracts* to facilitate enforcement against malicious acts in digital supply chains. Software is shared with licence buyers by using *Double Oblivious Protocols* and applying *robust watermarking* beforehand. Users are required to acquire software licences by signing smart contracts, which have a cryptocurrency as a collateral. In this manner, whenever they violate terms of agreement (i.e. distributing or altering the software) it is easier to impute responsibilities on in frictions and to enforce automatic and immediate fines.

# Chapter 7

# IOU Credit Networks

Talk by: Aniket Kate

Chapter by: Bernardo García Bulle Bueno and Gabriela Vargas Chacón

In this chapter we will study a concrete block-chain architecture which seeks to leverage on social trust to achieve resistance to Sybil attacks and higher scalability. This architecture is called IOU credit networks and was developed by a team in Purdue university. This chapter is mainly based on a talk by Aniket Kate, a member of such team.

## 7.1 IOU Credit networks

In IOU credit networks, transactors can be represented as nodes in a graph. Each node has few edges which correspond (ideally) to social connections of the agent, i.e. people whom the agent trusts and can reach socially outside of the network. When an agent wants to make a transaction with a node with which it is not directly connected, a path of debts is recorded from one transactor to the other, through their social connections. This yields a first advantage: debts can be simplified by collapsing them among agents.

I will now illustrate this. Suppose there are four agents: $A$, $B$, $C$, and $D$. Each one is connected to its neighbouring letters in the alphabet. This means $A$ is only connected to $B$ and $C$ is connected to both $B$ and $D$. Now suppose $C$ owes $B$ 100, let us ignore units for now. Now suppose $D$ owes $A$ 10. Since $D$ and $A$ are not connected, then the debt 'travels' through existing social connections through $C$ and $B$. Now, instead of $C$ remembering that she owes $B$ 100 and that 10 will pass through it from $D$ to $A$, the debts are collapsed. This means $C$ will only remember that it owes $B$ 110, and that $D$ owes it 10. Likewise, $B$ will remember $C$ owes her 110 and she owes A 10. This scenario is illustrated in figures 7.1 and 7.2.

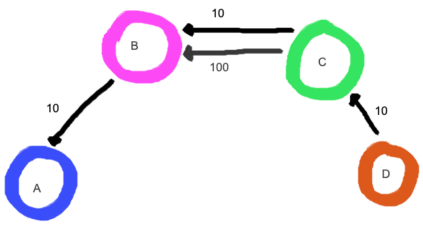


Figure 7.1: Illustration of debts without collapse

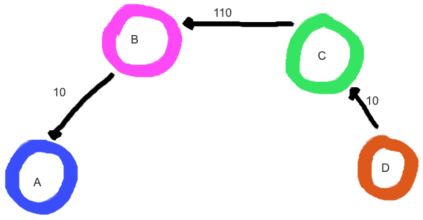


Figure 7.2: Illustration of collapsed debts

The reader may have realised that the problem of tracing routes in large graphs where each node has few connections can increase rapidly in terms of complexity. Before moving on to the advantages of the architecture, we will explain how this problem is bypassed.

### 7.1.1 Tracing paths

Since something important in this problem is scalability, landmark nodes are used to reduce complexity. Landmark nodes are regular nodes that are chosen when the set complies two conditions. First, there is a possible route between any landmark node and another. Secondly, there is a possible route between any node and a landmark node. All routes between such nodes are calculated, as well as routes between any node and a landmark. That way, if we want to connect a node to another, we only have to connect each to a landmark node. Then we connect them through a known route between both landmark nodes.

### 7.1.2 Transactions

The transactions in IOU are designed to be more scalable than in other cryptocurrencies. This is because, instead of a globally endorsed ledger, the system uses pairwise transactions made off the main system.

Suppose person A wants to transfer money to person B, through a path with several agents. Then person A engages in communication with the first agent in the path. Person A promises the amount of the transfer to that person, but that person must get a signature from the next node in the path. The signature can only be obtained if the first node promises the amount of the debt to the next person. This style is continued until it reaches the creditor. The creditor will then give a signature which will unlock the money for all, making them all keep their promises. These contracts will expire sequentially so as to enforce that no one loses their opportunity to charge the money. Since agents will need motivation, a small fee is promised to everyone in the path. Then after they get the money that was promised, they get a little extra with respect to what they paid along the path.

This system of transactions promises a smaller fee than that of current international transfers. Remittances can also reach the other side of the globe in seconds, which outperforms today's banking system. Most importantly since communication is made in a pairwise manner, rather than when trying to reach consensus on a global ledger, several transactions can be done simultaneously. This is a key factor as, if the world is going to move to a block-chain currency system, the amount of transactions will be far more larger than today. That will make it necessary to find ways to make the system able to process more transactions each time, such as IOU.

## 7.2 Sybil attacks

Sybil attacks refer to attacks on networks worked by creating a fake identities on a network. Consensus algorithms in distributed systems have their performance guarantees bounded by the number of malignant nodes. Therefore, if an attacker can create as many identities (nodes) as s/he wants, no distributed system can guarantee consensus. Other blockchain systems such as bitcoin are liable to such attacks, as proven in this publication [16].

How could a Sybil attack be made through IOU? Suppose we have connected agents B and C. B owes 100 units to C. A is connected to B and C is connected to D. Then, A and D report that D owes A 100 units. Then, by simplification, the debts change to D owing 100 to C, and B owing 100 to A. D will then default after having B pay to A. This way, B will tell C that it has already paid his/her debts, and C will remain unpaid. This example is illustrated in figures 7.3 and 7.4.

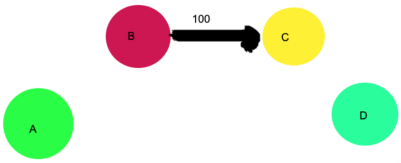


Figure 7.3: Illustration of debts before an attack by A and D

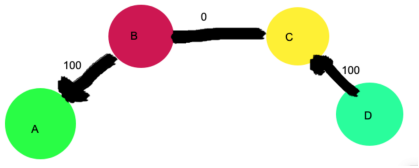


Figure 7.4: Illustration of debts during an attack where A and D report that D owes A 100 units.

Based on this example, what's needed for a successful attack is controlling two identities which are connected through innocent nodes where actual debts are in place. Then the forgers make the innocent debtors pay them their debts, and default on the innocent creditors. These attack

### 7.2.1 Tolerance to Sybil attacks

IOU has a natural resistance to such attacks. The influence a node can have in the whole system is limited by the edges it has. The edges in the architecture

are thought to be made only when there a social connection behind it. This social connection will carry as much trust as each user cares to enforce.

The limitation in terms of edges means that fake identities will face hardships connecting themselves to the real world. The connection of the fake cluster to the real world will probably be few nodes which are in fact the attacker, her/his accessories in the crime, and a few unlucky innocents. This is not enough. What's needed is that innocent debtors and creditors are caught as the only path between to malignant nodes (or clusters). If this case happens, then they are liable to an attack. However, the additional restriction of social trust works as a safeguard against those attacks. Moreover, the fact that there is a social connection between all of the nodes, both victims and perpetrators, outside methods of contract enforcement are possible.

This kind of system has already been successfully implemented in the past. Ostra, for example, uses it to prevent spam e-mails. SumUp uses to it count votes with resistance to Sybil attacks.

## 7.3 Proof of work

Bitcoin's proof of work system has become a large consumer of electricity worldwide [17]. This is because a proof of work needs to be obtained while mining. While this may be useful to the miners, it does not carry an intrinsic value. This means that energy is used only for the sake of it. IOU's approach does not make use of proof of work concepts. This further gives it an advantage over current mainstream systems.

## 7.4 IOU in the future

IOU has main advantages: it removes the need of global consensus and a ledger, it allows for fast transfer of funds worlwide, it can scale to more transactions per second easily, it leverages social trust to be more reliable, and it does not rely on proof of work concepts which waste energy. IOU represents an architecture with many uses in the future. Even if it is not used itself, some of it's ideas will surely contribute to the future use of blockchain in society.

# Chapter 8

# Atomic Cross-Chain Swaps

Talk by: Maurice Herlihy
Chapter by: Erick Chávez, Andrea Vargas

Let's imagine a situation where Alice has altcoins and wants a car, Bob has bitcoins and wants altcoins, and Carol has a car and wants some bitcoins. It sounds like they could make some exchanges and be better off getting what they want. In this section we will introduce *atomic cross-chain swaps* in order to make this exchanges possible. [18]

## 8.1 Atomic cross-chain swaps

An *atomic cross-chain swap* is a distributed coordination task where multiple parties exchange assets across multiple blockchains. An *atomic* swap protocol guarantees (1) if all parties conform to the protocol, then all swaps take place, (2) if some coalition deviates from the protocol, then no conforming party ends up worse off, and (3) no coalition has an incentive to deviate from the protocol.

The protocol for making this swaps possible use smart contracts in order to execute them when some conditions are met. In this case, we will use a *hashlock* which is implemented with a cryptographic hash function $H(\cdot)$. The hashlock $H(s)$ encrypt a secret $s$ and the contract specifies that the one who holds $H(s)$ will pay an asset to the person specified in the contract, if $s$ is provided, but if the secret is not provided before $t$ time elapses, the contract will be cancelled.

We define $\Delta$ as the time needed to publish and notice a contract. So in this case, we define a directed graph (digraph) $\mathcal{D}$ with vertex set $\mathcal{V} = \{A, B, C\}$ and edge set $\mathcal{E} = \{(A, B), (B, C), (C, A)\}$, which are ordered pairs of the vertex set, the direction of the edges represent the direction of the exchanges. This graph is represented in figure 8.1.

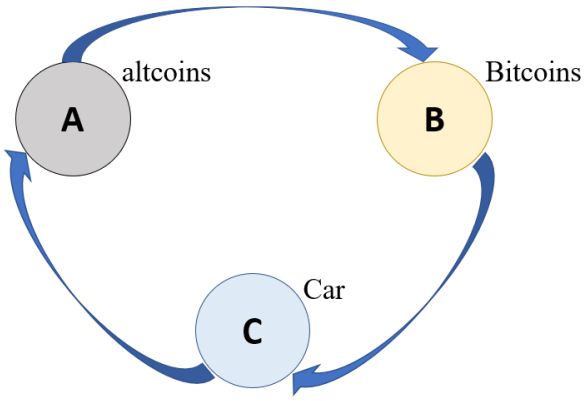


Figure 8.1: digraph of $\mathcal{D}$

In this example, $A$ can define a contract that will pay some altcoins to B if B provides the secret $s$ that produces $H(s)$, with a timeout of $6\Delta$, starting from $t$, in other words, before $t + 6\Delta$ (we will understand why we need $6\Delta$ in an intuitive manner). This contract is published at time $t + \Delta$. Remember that only $A$ knows $s$. Then, with the same logic, $B$ defines a contract that will pay some bitcoins to the first person that provides $s$ that produces $H(s)$, with a timeout of $5\Delta$. This contract is published at $t + 2\Delta$. Finally, $C$ defines a contract that will give the car to the first person that provides $s$ that produces $H(s)$, with a timeout of $4\Delta$. This contract is published at $t + 3\Delta$.

Now, in order to get C's car, $A$ provides $s$ to the contract held with $C$, before time $t + 4\Delta$, then $C$ can provide $s$ to the contract held with $B$ and get the bitcoins before time $t + 5\Delta$. Finally, $B$ can provides $s$ to the contract held with $A$ and get the altcoins before time $t + 6\Delta$. These differential of times between contracts guarantees that every participant will have at least $\Delta$ time to complete the transaction for their own good.

### 8.1.1 What could go wrong?

There are plenty of reasons why this exchange could go wrong, we will point out some of these.
1) Lets consider the contract defined in last section between *Alice*, *Bob* and *Carol*. What would happen if $Carol's$ network fails before releasing the car? well the contract would met the timelock and the contract would return the assets to the original owners. Not bad, everyone gets a refund and nobody loses
2) Now imagine that *Carol* gets $s$ and give the car to $A$, but before completing the transaction providing $s$ to the contract between her and Bob, Carol's network fails, then the contract would met the timelock making *Carol* unable to claim Bob's bitcoins, hence *Bob* would not make any exchange and *Alice* would have a surplus. No one will return the car back or give any money to *Carol* so

she would lose.
3) For this failure example, lets supose we are at time $t + 2\Delta$ where AB and BC contracts are both published, but CA is not. If at this point Alice accidentally reveals $s$ to Carol, s will become public and Bob will be able to get Alice's altcoins, similarly Carol will get Bob's bitcoin, but Alice would not get the car because none contract specifies it.

So as we can see, this protocol is not perfect. Now we will review atomic swap protocols in order to cope with this kind of problems.

## 8.2 Atomic Swap protocols

Now, we will analyze some topologies in which this kind of swaps are possible, and some problems present in systems with a wrong topology configuration.

### 8.2.1 Swap Diagraphs

Given a directed Diagraph $\mathcal{D} = (V, A)$ where each vertex represent a party and each arch represent a proposed asset transfer, the participants involved can follow different strategies to lead the swap into different outcomes. Swap is a cooperative game, but in order to get into some of this scenarios, we need to assume that some parties can form coalitions commiting to common strategies.

After the chosen strategy has been followed, each party will end up with one of these payoffs:

1. Deal: all archs connecting the vertex are triggered.
2. No Deal: none of the archs connecting some vertex were triggered
3. Free Ride: the party represented by this kind of vertex receive at least one asset, but none of the outcome archs get triggered. This means, the party get free assets.
4. Discount: all entring archs are triggered, but just some of the outcoming ones are.
5. Underwater: this is the worst case scenario, the party gives some asset but doesnt get any in exchange

If we wanted to order preference for these payoffs, this will be as follows shown in figure 8.2:

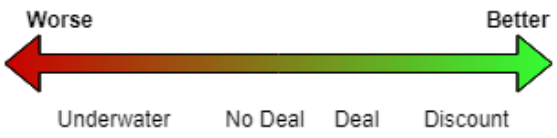


Figure 8.2: Payoffs

Once we explain the differt payoffs, lets define **Uniform Swap Protocol** properties.

- If all parties conform, they finish with payoff **DEAL**
- If any coalitions deviates, no conforming party finishes with payoff **UNDERWATER**

In order for a swap to be *atomic* the protocol needs to be *uniform*, but also be a *Strong Nash Equilibrium*, which means that everyone follows one strategy but if a coalition deviates, it would not improve its payoff

It is proved that if D is a Strongly Connected Directed Graph, then any uniform swap is atomic, so its also a strong nash equilibrium. But if D is not strongly connected, then no uniform swap is atomic

- **Strongly Connected**: Every vertex is reachable from every other
- **Feedback Vertex Set:** Set of key vertex that make the graph strongly connected.

# Chapter 9

# Blockchain Scalability via Layer 2 Protocols

Talk by: Aniket Kate

Chapter by: Didier Muñoz Díaz

In blockchain, probably the mostly discussed problem is scalability, refering in a wide sense to the ability of increasing its capabilities. In particular, increasing the transaction throughput (the number of transactions per second: TPS) is of primal interest and traditionally has been considered a layer 1 problem. This chapter tries to overview the reason for increasing scalability via layer 2 in contrast to layer 1 solutions.

## 9.1 Scalability in Layer 1

Scalablility in layer 1 (also called base-layer) usually works directly with the blockchain and therefore is greatly related to the scalability of the network. The main objective being to get the network do increasing number of tasks. This scalability can be seen in two ways:

- Vertical: more power can be added to the current network nodes.
- Horizontal: more nodes can be added to the network.

Both ways have their limitations. Let's consider we are trying to increase the TPS in a public blockchain as Bitcoin. In the case of *horizontal scalability*, more nodes does not mean more transactions per second processed because transactions are processed by every node in the network validating them and updating the ledger. This is the reason why most of the approaches choose vertical scaling.

Vertical scaling usually involves increasing the size of the block but this has its own drawbacks as is shown in the next sections.

### 9.1.1 Bitcoin in Layer 1

Some of the scalability related limitations of Bitcoin are as follow:

- High transaction-confirmation latency.
- Bitcoin is based on probabilistic consistency guarantees.
- Very low transactions per second (TPS) with an average of 3 to 7 TPS.
- A new block is added every 10 minutes.

Comparing these numbers to the $47K$ TPS required by Visa, we can see that there are still big improvements to be made in order to be able to compete with centralized solutions.

In an attempt to scale Bitcoin by increasing transaction throughput there would be two obvious solutions:

1. Increase the size of the blocks.
2. Decrease the block interval.

**Increase block size.** As currently designed, Bitcoin has a block size of $1MB$ meaning as much as 4200 transactions. If there is 1 block added every $10min$, then the transactions throughput is of 7 transactions per second.

If we were to increase the block size to $100MB$ then we could increase the throughput to 700 transactions per second.

There are reasons why this is not a good solution:

- Decreases fairness by giving large miners an advantage.
- Requires more space.
- Requires more network bandwidth.
- Requires more verification time.

**Decrease block interval.** In the second approach we could try to decrease the time interval to which a new block is added to the blockchain. In this way we could increase the transactions throughput by adding more blocks in less time. For example, if a new block were added every minute, then the number of transactions per second would be of 70.

Some of the drawbacks of this strategy are:

- Requires decreasing mining difficulty.
- Could lead to more forks.
- Can result in network instability (many branches).

### 9.1.2 Scalability in layer 1 drawbacks.

By vertical scalability, increasing the block size requires more powerful nodes, and therefore can cause the incidental "banning" of small computers from participating, resulting in a more centralized and insecure network. One could think that in order to address this issue, more validator nodes could be added, but this would provoke a decrease in speed (lower scalability).

In short, vertical layer 1 scalability cannot achieve all three speed, security and decentralization at the same time. In blockchain this concept is also known as the *impossibility triangle*, only two out of three can be achieved: scalability, decentralization and security (See Figure 9.1). Bitcoin achieves good decentralization and security by sacrificing scalability, while Ripple accomplishes desirable TPS and security but is considered a centralized solution.

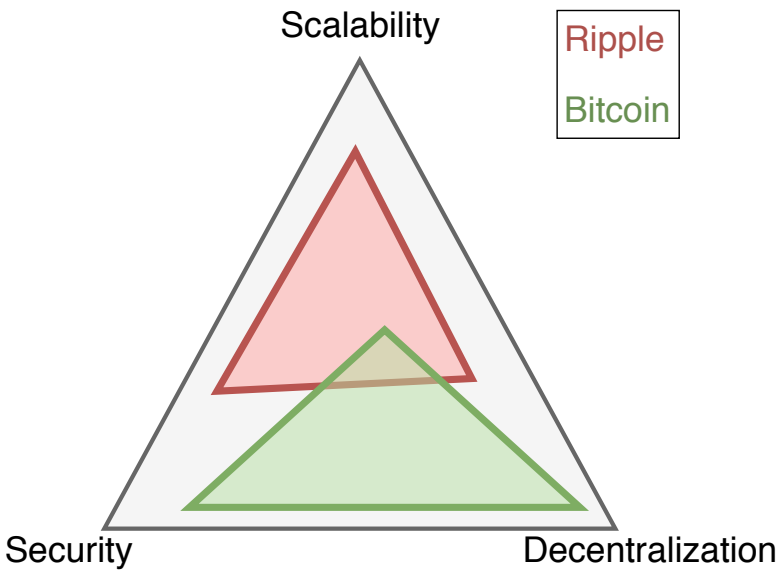


Figure 9.1: The impossibility triangle in blockchain.

## 9.2 Scalability in Layer 2

Consider having a secure base-layer blockchain. We would be sacrificing scalability by trying to keep the network decentralized. Another way of improving scalability while leaving the layer 1 as is, keeping almost the same security and decentralization, is layer 2. A layer 2 solution is one that is built on top of

another base-layer blockchain (Bitcoin, Ethereum, etc) by building new protocols. Note that it is important that the base-layer remains secure, because if the base-layer security fails, then the layer 2 would also fail (a secure protocol cannot be built on top of an insecure one).

The basic idea of layer 2 protocols is that you can improve scalability by letting interactions happen off-chain while reffering to the base-layer blockchain when necessary. In this way, transactions throughput can be improved by allowing fast off-chain transactions that eventually interact with the base-layer benefiting from its security.

### 9.2.1 Bitcoin in Layer 2. Lightning.

One of the layer 2 solutions for Bitcoin is the *Lightning Network*. This solution aims to solve the scalability problem by allowing fast off-chain transactions between participants.

The main idea is:

- Not all transactions are required to be recorded on the blockchain. Several transactions between same participants can be summarized by one final transaction. This also reduces the transaction commission that has to be paid.
- On-chain interaction (base-layer interaction) is used to establish payment channels (layer 2) between parties.
- Off-chain interactions must remain honest.

Considering 2 parties, the lightning protocol process is the following:

1. Create a bidirectional channel by generating a multisignature address (on-chain transaction). To open the channel each party sends money from the personal address. To spend the money in the multisignature address, both parties are required.
2. Off-Chain transaction.
   (a) To initiate an off-chain transaction, each participant generates a new key.
   (b) Each party creates a local transaction with the payment channel and the corresponding share.
   (c) To close the channel, both parties could agree on a final transaction that allow them to retrieve the corresponding money from the multisignature address.

In the end, this can be extended to more than 2 participants by allowing a network where two nodes are connected if they have an open multisignature channel. In this way one participant could send a transaction to a second one to whom no multisignature channel is open, by sending the transaction through a third participant connected to both. This transactions would all be made off-chain and therefore would be fast.

# Chapter 10

# Blockchain Privacy: Challenges, Solutions and Unresolved Issues

Talk by: Aniket Kate

Chapter by: Rodrigo Medina and Carlos Valenzuela

The success of cryptocurrencies helped block-chain technology gain widespread recognition as a reliable way of recording transactions in a non-centralised way. However, the public nature of block-chain represents a threat to users privacy[19]. This chapter focuses on privacy issues derived from block-chain technology applied to cryptocurrencies and some solution proposals. This chapter is mainly based on a talk by Aniket kate, in which he described current state of the art on those topics.

## 10.1 Privacy issues

In the last decade, blockchain technologies uprising caused turmoil in several economic activities, specially in financial systems. The rise of bitcoin showed that blockchain is a viable way to implement currencies, and more and more banks are adopting its technology to facilitate decentralised transaction storage systems. This is by no matter a consequence of chance. Blockchain offers the following advantages:

- *No third-party seizure*: Transactions happen peer-to-peer, just like hard cash. There is no need of a central authority regulating transactions, which eliminates the risk of seize or manipulation[20].

- *Transparency*: All transactions are documented in a public ledger, which allows all system users to check approved transactions at any given moment[20].
- *Theft resistant*: Stealing bitcoins is a difficult task, as transactions require a private key, which is usually kept offline, out of reach from malicious adversaries. Furthermore, once transactions are performed, they cannot be reversed[20][1].

Nevertheless, there is a serious drawback regarding the latter technology: Privacy concerns. Greenwich Associate's survey found that more than half of surveyed people cited transaction confidentiality as a major security concern [2]. Many people may ask why privacy is a big concern, as the line between public and private gets thinner this days. After all, there is people willing to publish their keys, making their transactions public. Nonetheless, privacy regarding transactions have its roots in tangible issues:

- Most people don't want their medical records being inferred by their purchases.
- Employees don't want the companies they work with to know whenever they buy products from competing companies.
- Competing organisations do not like to be snooped by others.
- In some regions, asset privacy may be a matter of life or death, as abundant resources turns you into a target for criminal organisations.

For that matter, privacy is enforced by legislation, regulation (client confidentiality) or contract (commercial confidentiality)[3], and it relies on centralised institutions for that. For example, Banks and financial corporations spend big shares of their resources in protecting the privacy of their clients.

Bitcoin gave the appearance of being an anonymous friendly, protecting its users privacy. This conception derived from the use of pseudonyms (addresses) and the ability to change them in each transaction, which in theory protected users identity. However, it's been constantly proven that with simply heuristics, it is possible to classify addresses which are more likely to belong to a determined user.

## 10.2 Key approaches for blockchain privacy

As mentioned above, blockchain technology offers improvements and solutions to a wide range of economic activities. How to benefit from them without sacrificing privacy? There is no single answer for that. Instead, several approaches have been proposed. Next, we explain key ones and their characteristics.

[1] Explanations of the advantages of bitcoin were taken from Mauro Conti's article[20], however Anonymity and lower transaction fees were left out as they may no longer be applicable benefits

[2] As stated by Aniket Kate in his talk.

[3] As stated by Aniket Kate in his talk

### 10.2.1 Mixing (Coinjoin mixing and third-party Mixing)

This method consists in taking a set of similar transactions (i.e. the transferred amount of bitcoins is the same) and randomly shuffling their signatures in order to make it harder for outside parties to infer the origin of the transaction. This method requires a previous agreement from the participants but coordination may or not be required, depending on the implemented solution. P2P solutions require that transactors coordinate beforehand, while single Mix-based solutions rely on a third party that automates this step. As it is a previous step to the transaction broadcast, it is perfectly compatible with bitcoin transactions. However, there is no privacy on the transacted values and it requires coordination/communications between participants.

### 10.2.2 Ring Signatures

This approach consists of grouping transactors and group them into a common key. Therefore, it is difficult to detect which particular key holder from the group performed a given transaction, because all key holders are equally likely to do so. As it requires an extra step, in which personal key signatures are converted into a group key signature, it is not compatible with bitcoin.

### 10.2.3 Stealth Addresses

This solution is focused on protecting receivers of currency. It requires senders to create random addresses (one-time use only) for every transaction.This makes different payments made to the same recipient untraceable as these transactions appear on the ledger as payments to multiple addresses. The owner of the private key can then use their private view key (shared with the sender) in order to see all incoming transactions [21].

## 10.3 Conclusions

table 10.1 summarises explained methods and their characteristics. It is important to mention that additional methods centred on making values of transactions private. These methods, like the Additive Homomorphic Commitments with Range Proofs rely on combinations of methods mentioned above and they usually one or more properties in order to protect the value of the transaction.

| Method | Relationship/sender anonymity | Amount/Value privacy | Co-ordination requirements | Bitcoin compatible | Examples |
|---|---|---|---|---|---|
| Coinjoin mixing (P2P) | Yes | No | Yes | Yes | Coin Shuffle, Coinparty, XIM, Coinshuffle++, |
| Single Mix-based Solutions | Yes | No | No | Yes | MixCoin, Blindcoin, TumbleBit |
| Ring signatures | Yes | No | No | No | Monero |
| Stealth addresses | Yes (recipient) | No | No | Yes | Monero, Z.cash. |

Table 10.1: Privacy enhancement methods

# Chapter 11

# Wallets and Custody

Talk by: Maurice Herlihy
Chapter by: Luis Alpízar

After reviewing the concept of blockchain and specially of Bitcoin from different sides of the distributed computing field, it is now important to consider 2 major features: the first, how to manage the money of someones account and the second, what are the legal consequences of this activity.

As discussed in the seminar, the management of a Bitcoin account requires a pair of keys that create a safe environment for transactions in the blockchain. These keys need to be stored in a safe place, therefore, as consumers of the technology we need particular wallets that will be described.

Also, because of the characteristic of ownership of a Bitcoin account, it is mandatory to indicate, at least briefly, the application of the Custody Rule that the Securities and Exchange Commission (SEC) of the United States of America applies in the securities industry. The SEC is well known as the agency that protects investors, maintains the fairness in these markets, and facilitates capital formation. Thus, it is relevant to consider how the Bitcoin and blockchain technology could be regulated.

### 11.0.1 Types of wallets

First of all, Bitcoin and other cryptocurrencies live inside the blockchain, this means their digital value must be exchanged to a physical value in order to use it in the real world.

These digital currencies are addressed by special keys which are kept in special wallets, since users do not have the opportunity to grab them as another belonging and treasure them in a particular place. The purpose is the same as any traditional wallet, to protect our belongings from theft.

As it was mentioned in the seminar, there are several ways to take advantage of the pitfalls that arise with the popular Bitcoin activity. One of them is hack-

ing, so in order to protect the currencies, one must decide over different types of wallets to secure the private information of these cryptocurrencies accounts.

Although the technology is new and innovations might transform this topic, several kinds of wallets were discussed as showed in the figure 11.1. The main classification of wallets are mentioned under the "General types" and the rest correspond to sub-types according to a given criteria.

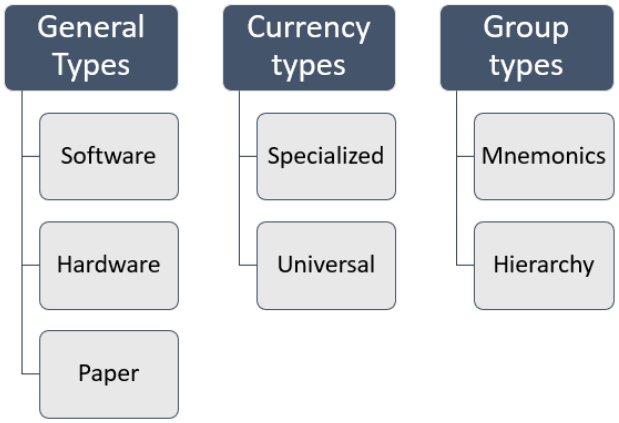


Figure 11.1: Types of wallets

1. Software. A software wallet consists of a program which can be downloaded to a desktop or mobile platform for online management. These wallets stand out for convenience, however, are highly vulnerable to virus and hack attacks. Several examples were presented, although the most popular might be the hacking through a Google Chrome extension and the use of "Hola" VPN.

2. Hardware. A hardware wallet adds an additional layer (referenced as an air gap) between a computer and the security of the keys. This means that the information is stored in files inside an electronic card or USB device. These wallets are more secure since they live in the physical world, however, they are useless without a computer. Once the wallets are plugged in, any harm can be done as the previous type.

3. Paper. A paper wallet is similar to a hardware wallet since it also lives in the physical world. The paper wallet is considered to be the most secure wallet. These wallets consist of a QR code for the public key and another one for the secret key. Even though this wallets does not need to be plugged in a computer, they do need to be pass through a scanner which puts some risk when entering the digital world. These wallets also have a drawback with colour blindness or carefree people since it is possible to publish the private key instead of the public one.

In addition, producers of these services are also creating specialised and universal wallets to offer in the market. As the name implies, the former aims to work with one type of cryptocurrency and the latter works with several currencies at a time. One must consider that transactions between different types of cryptocurrencies are not that common, so wallets mainly keep different types of currencies but might or not permit these transactions.

A different approach to produce wallets is offered with the use of mnemonics and hierarchies. These types add an additional layer of security by asking for different conditions in order to use the wallet. When using mnemonics, a multi-word tag generates the master key pair. If the phrase is misspelled, then the keys will not be valid. This type is normally used as back-up of an account. If a user lose the private or public key, he or she can generate the pair again with the mnemonic. When using hierarchies, a private key is divided into several parts that will be needed in order to validate a transaction. A rule will define how many of the parts are needed for a transaction. The one that gives or creates the keys is the one holding the master key, usually named boss. Only the boss can access all information of any pair of keys. This type is sensitive to collusion, so the groups might be looked after for prevention.

Apart from the kinds discussed, there are more technical wallets that store more than the keys of a particular account. These are called the full node wallets and miners use them because they build a copy of the entire blockchain. This means their data is trustworthy. Though, the full node wallet requires disk space and electricity to run all time in order to give a real time picture of all transactions. It also advertises the IP address of the owner, thus his or her information goes public which represents another risk to consider.

### 11.0.2 Custody Rule in the USA

Secondly, the possession of a Bitcoin or other cryptocurrencies can be directly linked to the owner. As a result, the owner has the choice whether to use or not an intermediary to manage a Bitcoin account. In one hand, the disadvantage of having a third party is the probability to have your money stolen or confiscated. At least, legal back-up might apply. In the other hand, carefree people can forget their account information and lose forever their assets. No legal back-up will be available.

When using an intermediary, the wallet is said to be custodial. If an American citizen is involved, then some legal notions must be considered. This means the Securities and Exchange Commission (SEC) should come into scene to apply the Custody Rule as in the securities industry[22]. According to SEC *an advisor who has possession of a client funds or securities and who has power to misuse them* is called custodian. Normally, an advisor has custody if he or she has the authority to withdraw funds from a client's account.

In this way, it is important to point out that an advisor of a custodial wallet can become custodian when managing Bitcoin accounts. However, these rules are still to be modernised since cryptocurrency is a different and emergent field.

As an example, some scenarios were described in the seminar. Although

some of them don't have a clear resolution whether the Custody Rule applies or not, they also indicate or relate to what normally happens in the securities industry:

- Similar to the mentioned industry, what must be consider as the "funds" and "securities" in the Bitcoin market? An advisor is not custodian if he or she never had access to a clients' keys? Can the keys be considered as the "funds" in the securities industry? Since keys are the pointers to the Bitcoins, are they more valuable so the "misuse" of the keys makes someone custodian? But keys by themselves have no value.

- A smart contract can be custodial if it indicates what to do in a particular scenario? Since the contract is not a legal entity, the author might be custodian? What if the author cannot control the account? What if he or she can modify the contract? All the details of the contract matter so it should be open source. It is not clear if the Custody Rule could apply because of the actions of contract itself or the possession of the keys.

- A key sharing strategy permits a transaction when $m$ out of $n$ keys are gathered. What are the implications for the Custody Rule in this situation? Normally the holders of the shards have no personal links and it is said the shards operate independently, so no custody is applied. But if not, then shard holders would become custodians since they can collude?

In conclusion, the popularity of cryptocurrencies will be determined by the technical and regulatory strategies developed to manage this kind of accounts. It will be mandatory that the rules of this activity get well defined and that every participant get to know what is permitted and what is not.

Without these rules, problems will still occupy the weekly news when dealing with Bitcoin or other cryptocurrencies in the blockchain instead of positive aspects of the technology. People need to trust the use of wallets in order to create a good environment for the Bitcoin and the blockchain practice.

We have seen that hackers are the main threat for this digital market so attacks might take less time and preparation than what is needed to rob a main bank for example. Therefore, security is a priority in this field.

# Chapter 12

# Open Problems Innovations and Challenges

Talk by: Amr El Abbadi
Chapter by: Edgar Granados, Luis Alpízar

This chapter describes the current challenges and innovations of Blockchain and its applications. As Blockchain is a technology that provides a secure and reliable mechanism when generating transactions between different parties, one of the major challenge faced is how to manage outside attackers such as hackers and cheaters.

A blockchain networks is intended to be flexible where different kinds of participants can interact. However, it is not fast enough to replace third parties when managing transactions because a transaction is a highly computational consuming operation (and therefore high power and time consuming). The reason for this is that for any transaction, there is a need to validate and save all the history of the Blockchain.

The problems presented above led to interesting and open questions such as: what if the people inside the network collude? Are faster transactions possible?

### 12.0.1 Attacks inside the blockchain

Blockchain is potentially vulnerable to a 51% attack. This attack can be achieved by malicious members that control 51% of the network by creating a double spending situation by forking the chain at any block. However, this attack has specific limitations e.g., in Bitcoin a transaction cannot be undone after it reaches six degrees deep in the blockchain.

Although reaching 51% control of a blockchain network is not easy to achieve, an alternative attack only requires control of 33% of the network. The attack can be exemplified as follows: a network is composed of only three miners: 2 honest and one selfish miner. If the selfish miner finds a new block but does not

announce it to the other two miners, he can start mining immediately though gaining an advantage. While the other two honest miners are trying to find the next block, they are wasting their resources as they will not be rewarded. When the honest miners find and announce the new block, the selfish miner has two possibilities: announce two blocks and take the reward (has been mining for longer) or create a fork by announcing a new hidden block and expect the other miners to split. If the split happens, this will lead to having 2/3 of the miners in the selfish miner block, therefore, increasing the probability of finding the new block in the one of the selfish miner.

### 12.0.2 The Lightning Network

The *Lightning Network* represents a novel alternative that enables instant payments across a decentralised network of participants by using smart contracts.

In the second place, a different way to make a transaction is presented with the lightning technique which uses smart contracts to enable instant payments across a decentralised network of participants[23]. The main aspect of such transactions is that they must be seen as created outside the chain while relying on a payment channel inside the chain. The critical challenge is to prevent participants on cheating each other.

To prevent cheating, the first step is to create a multi-signature transaction. Then it is necessary to fund it with another transaction which opens a channel between the participants when published on the blockchain. Each participant offers a specific amount in the transaction, but no one will sign it.

The next step is to each create local transactions with the previous multi-signature transaction as the input, some benefit as the output and the condition to meet will be if one has the other secret key. The expected output is the proportional part presented by each one in the multi-signature transaction which will give both sides the security of their funds if they suspect cheating form the other side. It is essential to include a time condition: the side broadcasting the transaction without the external key will need to wait a considerable number of blocks to be appended in order to retrieve their money. The time condition prevents anyone from broadcasting without consulting it previously.

The lightning protocol settles a safe environment for both sides; however, this is not the purpose of the lightning network. After this, participants can create several transactions which will update the balance of the multi-signature until both sides agree to stop. These transactions can be seen off-chain, and no harm will be done since each transaction gets a new pair of keys. To close the channel, one participant has to broadcast the last exchange or mutually agree to close the channel by publishing the open and final transaction on the blockchain.

### 12.0.3 Challenges and conclusions

While there is some criticism of blockchain, it is essential to point out that stronger critics come from outside the field. In particular, people are concern

about the cost of the resources taken in order to maintain and update the network. In this modern age where computer development has reached a point where energy care is vital, this new technology is coming in with a helpful solution to facilitate transactions and other activities in a completely open manner at the high cost of significant computation power. However, this has not been well acknowledged.

Because the concept is new and popular, the blockchain technology is being tracked by many enthusiasts from different fields who can benefit from it in different ways. This enthusiasm creates the need to care about the security in both ways inside and off the chain. As discussed before, attacks can come from different situations like analysing the code of a smart contract, using hacking techniques to steal digital wallets or even using scam skills in the physical world to convince people to invest without knowing anything about the blockchain concept.

Lack of rules and privacy can be exploited by malicious agents since much information is public in the network. A regulation could eventually take care of most of the negative aspects; however, it will take time to establish the optimal legal framework.

To sum up, this chapter can conclude that the scalability problem of the blockchain is a priority for the enthusiasts in order to speed up the transactions of this technology but so are the security mechanisms inside the network. Although the blockchain seems unlikely to be hacked entirely, some mechanisms can benefit malicious agents to take advantage over honest agents. While blockchain aims to promote inclusion, it is essential to remark that its electrical consumption worldwide is more than 150 countries together: there are other complex problems can outside the field. The electric consumption problem might be a critical challenge since it represents a severe dilemma to the future and continuous use of blockchain.

# Bibliography


[1] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1982.

[2] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[3] Jae Kwon. Tendermint: Consensus without mining. *Draft v. 0.6, fall*, 2014.

[4] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[5] Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. Parblockchain: Leveraging transaction parallelism in permissioned blockchain systems. *arXiv preprint arXiv:1902.01457*, 2019.

[6] Small business retail. https://usa.visa.com/run-your-business/small-business-tools/retail.html.

[7] Ittay Eyal, Adem Efe Gencer, Emin Gun Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, Santa Clara, CA, 2016. USENIX Association.

[8] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 279–296, Austin, TX, 2016. USENIX Association.

[9] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open

blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 17–30, New York, NY, USA, 2016. ACM.

[10] Samuel Falkon and Samuel Falkon. The story of the dao - its history and consequences, Dec 2017.

[11] Eric Banisadr and Eric Banisadr. How $800k evaporated from the powh coin ponzi scheme overnight, Feb 2018.

[12] J Buurman. *Supply chain logistics management*. McGraw-Hill, 2002.

[13] James Aitken. Supply chain integration within the context of a supplier association: case studies of four supplier associations. 1998.

[14] Easwar Vivek Mangipudi Pedro Moreno-Sanchez Aniket Kate, Mahimna Kelkar and Krutarth Rao. Dust-bt: Preventing supply chain tampering using blockchain technology, 2017. Poster at COSN'15.

[15] Khaled Salah, M Habib Ur Rehman, Nishara Nizamuddin, and Ala Al-Fuqaha. Blockchain for ai: review and open research challenges. *IEEE Access*, 7:10127–10149, 2019.

[16] George Bissias, A Pinar Ozisik, Brian N Levine, and Marc Liberatore. Sybil-resistant mixing for bitcoin. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 149–158. ACM, 2014.

[17] Andrew Miller, Ari Juels, Elaine Shi, Bryan Parno, and Jonathan Katz. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy*, pages 475–490. IEEE, 2014.

[18] Maurice Herlihy. Atomic cross-chain swaps. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 245–254. ACM, 2018.

[19] Tim Ruffing and Pedro Moreno-Sanchez. Mixing confidential transactions: Comprehensive transaction privacy for bitcoin. *IACR Cryptology ePrint Archive*, 2017:238, 2017.

[20] Mauro Conti, E Sandeep Kumar, Chhagan Lal, and Sushmita Ruj. A survey on security and privacy issues of bitcoin. *IEEE Communications Surveys & Tutorials*, 20(4):3416–3452, 2018.

[21] mycryptopedia. What are stealth Addresses? https://www.mycryptopedia.com/everything-need-know-stealth-addresses/, 2019. [Online; accessed 07-May-2019].

[22] Custody of funds or securities of clients by investment advisers. https://www.sec.gov/rules/final/ia-2176.htm.

[23] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.