POLITECNICO DI TORINO

Master's degree in Computer Engineering

Master's Degree Thesis

Enhancing Circular Bioeconomy using Distributed Ledger Technologies

Supervisor

Prof. Guido Albertengo

Company Tutors

Paola Dal Zovo

Andres H.M. Herrera

Candidate

Lorenzo Limoli

A.Y. 2020/2021

Abstract

In recent years, global sustainability challenges, such as world population growth and over-consumption of non-renewable resources, are pushing societies to establish and follow circular and regenerative approaches.

Bioeconomy and **circular economy** can help to alleviate environmental problems caused by the lack of modern and smart waste management systems, which do not provide operational transparency, traceability, security and trusted data provenance features. In fact, today's technologies leveraged for waste management, strictly relies on manual and centralized silos making them vulnerable and the single point of failure of the entire system.

To meet the challenge, the transition plan must consider the necessity to integrate economic and environmental perspectives as well as stimulating social awareness and dialogue, leading to more conscious behaviors. Organic resources such as food wastes or biodegradable and compostable packaging, can be transformed into organic fertilizer or in organic gas that can be used as fuel. However, currently they are not optimally utilized.

Waste resources are complex materials that are strongly varying in terms of composition, quantity and quality. To get high value-added bioproducts it is needed to ensure the quality of the inputs.

In the thesis is reported how **Distributed Ledger Technologies (DLTs)** could play a key role in enhancing the transformation process and provide important features that today's digital infrastructures are missing. This would help the transition towards scalable and decentralized systems with the capability to offer the mentioned features.

Finally, a **proof-of-concept (PoC)** applied to a circular bioeconomy scenario, and based on the novel kind distributed ledger of the **IOTA** platform, **the Tangle**, is proposed.

Table of Contents

Chapter 1

Introduction

Decentralization is one of the main purposes that currently drives the majority of the existing cryptocurrencies. Since the launch of `Bitcoin` in 2009, decentralization is the main ingredient used to build a peer-to-peer (P2P) financial system managed by no central administrators (such as banks for the traditional financial systems). Every cryptocurrency that has been designed after `Bitcoin` takes inspiration from it either directly or indirectly. In these kinds of financial systems, transactions are now stored in what are called **Distributed Ledgers** that are maintained by the peers instead of the traditional databases that are instead handled by a single authority.

1.1 What is a Distributed Ledger ?

A *Distributed Ledger* is a system aimed to **store**, **share** and **sync** digital data geographically spread across multiple sites[1]. Thanks to its distributed nature, it has no *single point of failure* because each node of the network contributes to avoid data-loss or data corruption. Another property that characterizes a *DLT* is the **immutability** of the stored data: once a particular data is added to the ledger, it cannot be updated or deleted instead it is meant to live there forever. A *DLT* enables decentralization by relying on a **peer-to-peer network**: each node stores (usually) the entire ledger, and once new data are received and added to it, the node has the duty to forward these updates of the ledger to its neighbours (`Figure` 1.1). Each peer (node) is allowed to add new records inside the ledger, but they are only accepted when everyone in the network agree the records meet every requirement of the ledger, for example having a valid digital signature. *DLTs* share characteristics with other distributed systems such as **Distributed Database Management Systems**.

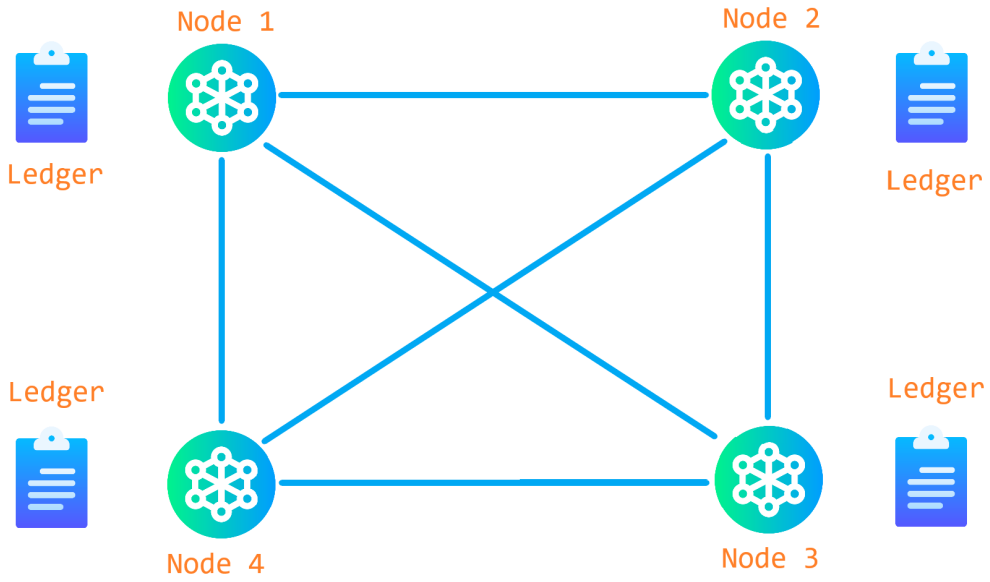

Figure 1.1: Distributed Ledger graph

Here there is a list of the main similarities:

- **Data Store:** the aim of both systems is to store data.
- **Data Share:** both systems make data available to users for processing.
- **Synchronization:** because of the intrinsic meaning of *"distributed"*, both nodes of *DLT*s and *DDBMS*s must be synced in order to maintain a consistent state of data.
- **Fault tolerance:** being distributed means also avoiding a single point of failure, as we mentioned before.

But there are also differences. The biggest one is about **trust**. We can identify two different aspects that are related with the concept of trust:

1. **Inner trust.** Can a node that contributes to the health of a distributed system trust the information received by its neighbours?
2. **Outer trust.** This category instead focuses on the users of a service that a certain system offers. Can they trust the data that the system provides? Can they have guarantees about a secure and correct use of personal data that they eventually have provided to the service?

As already explained, common *DDBMS* have to be managed by single entities. Entities are assumed to maintain all the nodes of their distributed system in

order to make it work correctly and efficiently. With this architecture, each node has no reasons to doubt that its neighbours would act dishonestly because all of them *live under the same rough*, so synchronization of data happens without trust issues. However, consensus mechanisms among nodes are needed to handle data consistency and fault tolerance. On the other hand, problems arise when there are users that want to benefit from a certain service provided by a certain entity (such as companies, governments or banks). In these cases users intrinsically have to trust the specific entity that provides the chosen service; moreover, they have to believe that this entity will properly handle their personal information that is eventually required.

Thanks to their peer-to-peer architecture, *DTL*s are instead managed by the simultaneous contribution of every single peer that participates to the network. Differently from *DDBMS*s, this architecture is based on the fact that **a peer doesn't trust at all the other ones**, so each peer of the network must verify each incoming data independently. Because of that, a more complicated consensus mechanism is needed for these kinds of networks: to bring all the copies of the ledger to a common and verified state, the majority of the peers must reach an agreement voting for the same decision. The mechanism influences also how users perceive a service *DLT* based: when a user decide to benefit from a certain service, is sure that every information that the service provides, has been verified by every node of the network independently and has not been altered by anyone by design. In addition, no more central authorities will hold users' personal data.

	DLT	**DDBMS**
Operations	Create - Read (Data are immutable)	CRUD (Data are mutable)
Architecture	Peer-to-Peer (Decentralized)	Hierarchical (Centralized)
Inner Trust	Each peer verify independently each incoming data because doesn't trust its neighbours	Each node trust its neighbours because they belongs to the same entity manager
Outer Trust	Users are sure that data are verified and unaltered	Users have to trust the validity and correctness of received data without guarantees

Table 1.1: Differences between *DLT*s and *DDBMS*s

Cryptocurrencies are today the most common use of *DLT*s. They are digital assets that use cryptographic techniques, to make transactions secure, and also *Distributed Ledgers* to store transactions into them. In this way a traditional

trusted third party (e.g. banks) is no longer needed to enable money exchange.

A *DLT* can be implemented using various architectures and methods. The most used is undoubtedly the **blockchain** and the most famous cryptocurrencies that use this technology are `Bitcoin`, `Ethereum`, `Cardano` and many others.

1.2 Blockchain

The term *"blockchain"* has been used for the first time in 2008, when an unknown person (or unknown group of people), with the alias of Satoshi Nakamoto, has published the so-called ***White Paper*** where it is explained how `Bitcoin` works.

As the term suggests, a blockchain is a data structure composed of a growing list of records, called **blocks**, that are linked together using cryptographic relations. Each block contains a **cryptographic hash** of the previous block, a timestamp and the transaction data (`Figure` 1.2) that are generally represented as a `Merkle Tree`, a data structure that we will discuss later on in this document. The *timestamp* proves that the transaction data existed when the block was published in order to get into its hash[2]. Each block is linked to the previous one to reinforce the entire chain: in fact, to be able to modify an already existing block within the blockchain, all the previous blocks must be altered too. These backward links between blocks are usually referred to as **hash pointers**.

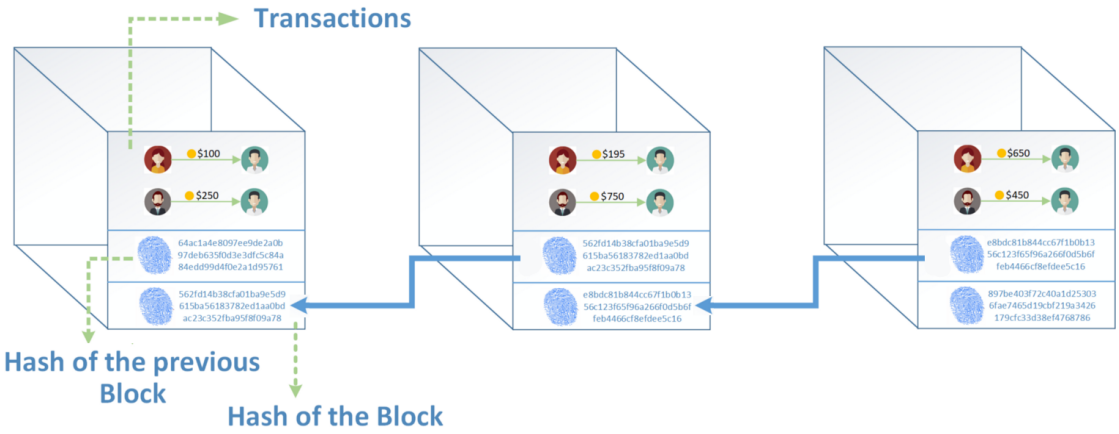

Figure 1.2: Blockchain Structure[1].

[1]Source: linnwealth.com

1.2.1 Structure

A blockchain is a **decentralized**, **distributed** and usually **public** digital ledger that is used, as we already explained, to record transactions across many computers so that each block that composes the entire blockchain, cannot be altered retroactively[2]. Every block is authenticated by **mass collaboration** powered by **collective self-interests**. Such a system guarantees by design a robust workflow and removes the characteristic of **infinite reproducibility** from a digital asset: it confirms that each unit of value is transferred only once, solving the long-standing problem of the **double spending attack** where a digital cash for the only fact of being *"digital"*, could be copied and so spent more than once.

Logically, a blockchain can be represented as a stack of layers that are from the bottom (`Figure` 1.3):

1. **Infrastructure:** the hardware used from the peers.
2. **Network:** all the mechanisms implemented to guarantee decentralization, availability and consistency through nodes communication.
3. **Consensus:** protocols that establish the rules that peers have to follow in the system to guarantee the health of it.
4. **Data:** the set of data stored in the distributed ledger such as transactions.
5. **Application:** e.g. dApps and smart contracts.

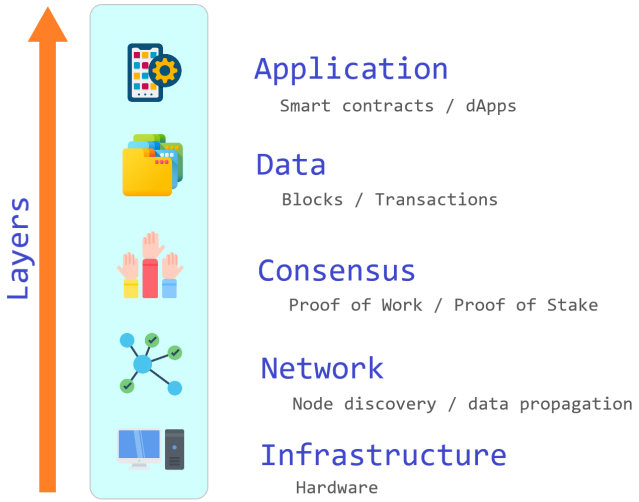

Figure 1.3: Blockchain Layers

1.2.2 Blocks

Blocks are sets of valid transactions and each block is linked to the previous one through the hash pointers. This iterative process of adding new blocks to the last valid one, confirms integrity and immutability of the previous blocks, all the way back to the initial one, named **genesis block**[3].

Even if the *"real"* blockchain consists of a linear chain of blocks, sometimes separate blocks can be produced concurrently. This phenomena leads to the creation of a temporary **fork**. To deal with this behaviour, that would compromise the security of the system, any blockchain has a specified algorithm for scoring different versions of the history so that one with a higher score can be selected over the others. Peers usually store only the history with the highest score, eventually overwriting the older version.

1.2.3 Blockchain policies

After the introduction of the original concept of blockchain, new policies have been devised about letting new nodes join the network.
Three different macro-types of blockchains can be identified with respect to chosen policy:

- **Permissionless blockchains.** Also known as public blockchains. The most striking example is Bitcoin again. Permissionless blockchains represent the original concept of "blockchain" thanks to its intrinsic nature: they are totally open to each user that wants to join the network with the aim of guaranteeing the purest ideals of decentralization. This implies that these kinds of blockchains are not owned by a single entity but instead, they belong to the community that believe in the project. Indeed, they are considered *community-driven*. Permissionless blockchains also guarantee free access to the data stored in the ledger that, in any case, are encrypted to provide a minimum level of privacy.

  Frequent concerns about public blockchains are about the concept of scalability, or in other words the ability of a system to improve its speed and efficiency as the number of the participants increase. In fact these systems are often not fully scalable because, even if the security and stability of it increases as the number of nodes increases, the number of transactions per seconds tends to remain constant or even to decrease.

- **Permissioned blockchains.** Also known as private blockchains. They are usually designed for business oriented solutions, or in other words designed

for companies. Contrary to permissionless ones, permissioned blockchains belong to specific entities. These kinds of systems introduce the concept of **governance and centralisation** in a network that was originally thought as fully decentralized and distributed[4]. Each entity acts as supervisor of the distributed ledger, deciding which user is allowed to enter the network and which message is approved. Instead of relying on the users' nodes, the consensus is entrusted to a small number of trusty nodes selected by the owner of the blockchain. Private blockchains also enable roles selection for each node of the network.

In this case, scalability is not a problem at all, instead thanks to the small number of selected nodes that manage consensus, the ledger state synchronization and propagation are way quicker. Of course the big compromise remains the centralisation.

In the end, new types of consensus protocols are introduced in the context of private blockchains such as the Proof of Authority[5] that, unfortunately, will not be discussed in this document.

- **Hybrid blockchains.** As the name suggests, it is a mix of the previous two, trying to get the best part of both. Ideally, when we talk about hybrid blockchains, we talk about freedom and controlled access at the same time. As usual, hybrid blockchain architecture is entirely customizable. The hybrid blockchain members can decide who can participate in the blockchain or which transactions are made public. This brings the best of both worlds and ensures that a company can work with its stakeholders in the best possible way[6].

1.3 Consensus protocols

We mentioned more than once that a distributed system needs, due to its architecture, a mechanism that allows for synchronization among nodes and validation of data. These mechanisms, also known as **consensus protocols**, are fundamental for the correct functioning of the system, especially for a DLT where the concept of trust becomes the core part to enable **decentralization**. Blockchains are no different, and since the release of Bitcoin, consensus protocols became an active research area.

1.3.1 The Byzantine Generals Problem

Before introducing how the most famous protocols work, it is necessary to understand where the need, to agree on certain decisions asynchronously to reach

consensus, come from.

The problem of *Byzantine Generals* was proposed in 1982 and it is a logical dilemma that shows how a group of byzantine generals would have communication problems when they try to reach an agreement for the next move[7].
The dilemma supposes that:

1. Each general has **its own army**.
2. Each army is **placed in a different location** around an hypothetical city to conquer.

We can now define the rules that each general must follow:

1. A general must choose if its army has to **attack or to retreat**.
2. Once the decision is taken, it **cannot be changed**.
3. It doesn't matter if the armies attack or retreat as long as every one of them **takes the same decision** to avoid to compromise the operation.

Communication problems are related to the fact that each general can communicate to another one only through messages delivered by messengers. This makes the entire problem really hard to solve. In particular:

- Messages could have delays or, worse, could never be delivered.
- Even if a message is correctly delivered, one of the generals could act dishonestly, for whatever reason, and send fake messages to confuse the others, causing the entire operation to fail.

But how any of this would be related to the blockchain? In this context we can imagine that each general represents a node (peer) of the network, and each one of them must reach an agreement on the actual ledger state. In other words, the majority of the peers must reach the consensus and execute the same action to avoid data inconsistency. This implies that the only way to reach consensus in these distributed and decentralized systems is to have at least $\frac{2}{3}$ of honest and reliable nodes within the network. Instead if the majority of the nodes decide to act dishonestly, the system becomes susceptible to malfunctioning and cyber-attacks.

1.3.2 Byzantine Fault Tolerance Algorithms

The Byzantine Fault Tolerance (BFT), is the property of a distributed network to reach consensus even when some of the nodes in the network fail to respond or respond with incorrect information. The objective of a BFT algorithm

is to safeguard against the system failures by employing collective decision making (both correct and faulty nodes) which aims to reduce the influence of the faulty nodes[5]. In other words BFT algorithms are those capable of solving the *Byzantine Generals problem*.

The most used consensus protocols that will be explained in this section are the **Proof of Work** and the **Proof of Stake**.

1.3.3 Proof of Work

Proof of work (PoW) is a form of **cryptographic zero-knowledge proof** in which one party (the prover) proves to others (the verifiers) that a certain amount of computational efforts has been done. Verifiers can subsequently confirm this expenditure with minimal effort on their part[8].

In other words the protocol imposes the execution of some tasks to the users of the service. These tasks are basically cryptographic puzzles characterized by two main properties:

1. They must be **complex enough to be solved** (so that a certain amount of computation power is required).

2. They must be **easy to verify** once the solution has been found.

Thanks to this mechanism, it is a protocol that is able to discourage denial of service attacks and other service abuses such as spam in the network.
The concept of PoW was invented in 1993 and, in fact, its original goal was to solve these problems instead of the ones related to consensus in a DLT discussed so far.

However, PoW became popular only after the `Bitcoin` release, turning into a foundation for consensus in permissionless blockchains and cryptocurrencies, in which miners compete to append blocks and to mint new currency.

Why miners are needed?

We already talked about the concept of trust in a distributed ledger context, so it should be now clear why consensus protocols must be used. In the PoW, trust is achieved by computational efforts done for the resolution of the cryptographic puzzle: once the solution has been found, the `solver` (the node who actually found it) communicate it to the other nodes in the network so that they can verify that it is correct. When the consensus is reached, then everyone trusts that set of transactions (the block) is valid, and it is pushed into the blockchain. Every node

that tries to solve these puzzles is called `miner`. Miners compete against each other to find the solution of these puzzles and so to add as many blocks as possible to the blockchain. But why should miners use their computational power to compute PoWs? What advantages do they get for lending their resources? The answer is simple: they get paid for their effort each time a PoW is computed. The more power you get, the more probabilities to generate the PoW you have, and the more PoW you compute, the more you earn. The process can be simply explained with this example:

- *Alice* wants to pay *Bob* with a certain amount of a cryptocurrency blockchain based. So she have to add this transaction in the ledger.
- Before the transaction is added, it must be verified. So each miner of the network starts computing the PoW as fast as he can.
- Once a certain miner solves the puzzle, the others check if it is correct. If the consensus is reached, then the transaction become valid and it is added to the ledger.
- The miner that found the solution is then rewarded for his effort. In fact, when a PoW is computed, new currency is minted and used both to pay the miner and to increase the circulation. Another source of income are **fees** that the *"transaction creator"* will promise to miners for the computation of the PoW (we will explain more in detail this mechanism when we will talk about `Bitcoin`).

1.3.4 Proof of Stake

Proof of Stake (PoS) protocols are a class of consensus mechanisms for blockchains that replace the competitions among miners to solve PoWs into a system where the network nodes, called `validators`, guarantee the validity of transactions by *staking* a certain amount of their cryptocurrencies.

PoS has been designed to avoid the huge energy waste derived from miners' competition. To have chances to be the first one solving PoWs you always need more and more computational power, and more computational power means an improvement of speed of PoWs resolution. This is what makes systems PoW based very secure and reliable because these costs that miners have to meet, make the event that they could hurt the network highly unlikely.

Another problem that PoS tries to overcome is the continuous decrease of decentralization due to the mining pools: today miners are grouped under organizations and companies with huge financial resources that allow mechanisms to cooperate

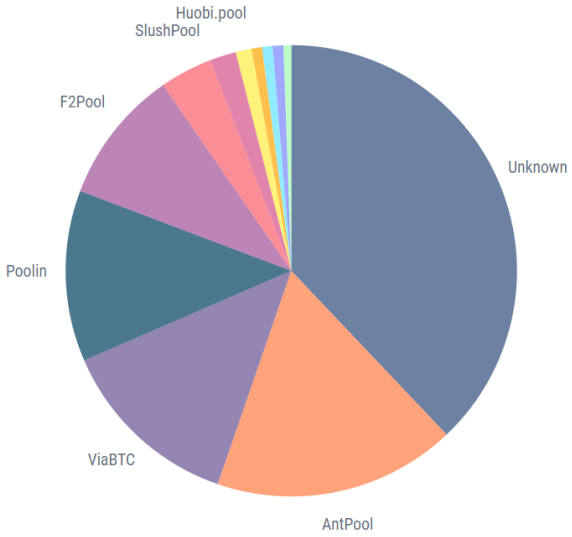

Figure 1.4: Mining pools distribution[2]

as one single node (and so as a single supercomputer) to solve faster each PoW. Each of these groups is called `mining pool`. With this method, all the incomes of this single node are distributed among the members proportionally to the provided computational power, rewarding also members that would have not solved a single block due to their low power. This is surely an advantage for miners, but the concept of decentralization becomes gradually weaker and weaker.

How it works?

PoS uses a mechanism that randomly chooses a validator. Contrary to PoW, blocks are not mined but *forged* instead. A node becomes a validator only when it stakes a portion of cryptocurrencies within the network, as a guarantee that it will act honestly for the network's sake (and for its own), otherwise they lose what they staked. Obviously it's not possible to spend or withdraw the resources once they have been staked.

The random selection algorithm bases its choice on different factors, guaranteeing that will be selected either the nodes with huge deposits and the ones with smaller

[2]Source: blockchain.com

ones, but still reliable. The principal variables taken into account are the size of the deposit (how much cryptocurrencies have been staked), the coin age (how long the stake is active) plus an ulterior random factor. The nodes with bigger deposits and with a higher coin age will be considered more reliable and will have of course more probability to be chosen as validators. It's important to note that, to maintain a fair probability distribution of being chosen among nodes, once a validator forges a block, its coin age is reset and before it can be re-chosen, a certain amount of time has to pass. The validator must check whether each transaction of that block is valid, then it signs the block itself and pushes it into the blockchain. Once a new block is added, the validator is rewarded with a fee deducted from the cryptocoins sent with the transaction. Nodes can't withdraw earnings before the entire network reaches consensus.

1.3.5 A comparison between PoW and PoS

At this point we can clearly notice how in systems PoS based, each node is discouraged from acting against the network because of the stake that would be gradually lost in case of dishonest behaviours; moreover when a node approve fraudulent transactions, it loses the right to be selected as validator in the future. The only way to bypass the security controls would be to own 51% of the network's liquidity, which is a highly unlikely hypothesis. On the other hand, as we said before, in systems PoW based, security and trust rely on the large use of both economical and financial resources[9]. This makes PoW less eco-friendly than PoS.

At first glance, PoS protocols seem to facilitate earnings to the nodes with the highest amount of resources, making them richer and richer. However, thanks to the random selection algorithm that we discussed before, PoS guarantees a fair and well balanced mechanism to manage the validators shifts.

With PoS, better solutions for scalability, decentralization and democratisation are provided: despite the huge computational power that mining pools provide to the network, the cryptographic puzzles usually require a great amount of time to be solved. This implies that there is some latency between the transmission of a transaction and its validation. The slow validation speed could be considered as a barrier of entry in traditional financial systems for all cryptocurrencies that adopt PoW as consensus protocol. On the contrary, in PoS either the selection of the validator and the validation of a block are performed in a few seconds enabling the support for real-time electronic payments.

	PoW	**PoS**
Scalability	Limited by the complexity of the cryptographic puzzles	Scalable by design: security is guaranteed by the stake and not by computational efforts
Transactions per seconds	Few transactions per seconds	Larger amount of transactions per seconds
decentralization	Affected by mining pools	Fully decentralized
Earnings	Facilitate miners with great computational power	Facilitate nodes with great financial resources on stake

Table 1.2: Advantages of PoS over PoW

Chapter 2

Bitcoin: The First Blockchain Implementation

During the last decades, the world's economy is clearly moving towards digitalization to make the financial systems more secure but mostly easier and more immediate. The virtualization of financial tools is nowadays very common, especially when we talk about money exchange made through mechanisms that just a few years ago would have been considered unfeasible. When people use credit cards or other payment methods, each transaction doesn't require an actual transfer of the physical assets (real coins or cash), instead it is represented as a virtual exchange of digital assets that occurs through digital processes that manages and tracks electronic money that are a representation of the real ones. This concept lays the foundation for the creation of the first cryptocurrency the world has ever seen, and the experiment is now considered one of the most successful ones in this compound.

The story of these virtual coins begins in 1983, when a certain cryptographer David Chaum developed a cryptographic system called **eCash**. Twelve years later, he developed another system, **DigiCash**, that used cryptography to make economic transactions confidential. However the term "cryptocurrency" was coined in 1998 by **Wei Dai**. In that year he was thinking about developing a new payment method that used a cryptographic system and whose main characteristic was decentralization[10]. Starting from Dai's idea, cryptographers started to get interested in the cryptocurrency concept and it has been hypothesized that, once established the rules of generation, realizing a coin that would be minted and distributed on the internet would not be impossible. This brought to the realization of `Bitcoin`, the first and most famous cryptocurrency that is now internationally recognized as an "alternative coin". From now on, will follow a great number of

alternative cryptocurrency that take inspiration from `Bitcoin` and that aspire to the same success.

2.1 What Bitcoin is and how it works

`Bitcoin` (₿) is a cryptocurrency created by **Satoshi Nakamoto**. He designed and published the idea in 2008 in a document called White Paper with the title *"Bitcoin: A Peer-to-Peer Electronic Cash System - Satoshi Nakamoto"*[11]; then he developed and released the first build in 2009. `Bitcoin` is an "anonymous" and distributed digital coin managed by a peer-to-peer network where its price is tuned only from supply and demand. The main purpose for which Nakamoto wanted to introduce a new and alternative financial system through `Bitcoin`, was to **contrast the global economic crisis** that began more than one decade ago, including America's superpower. The primary consequence of this crisis was the devaluation of the Fiat currencies (i.e. USD, EUR etc) due to inflation. Thanks to a distributed system such as `Bitcoin`, central authorities would not be in charge of making decisions by themselves that would influence the world's economy anymore.

We mentioned more and more that `Bitcoin` uses the blockchain, and so it relies on the concept of trust enabled thanks to the use of cryptography and in particular of digital signature: Nakamoto himself states that *"we define digital coin as a chain of digital signatures"*. A proper use of these cryptographic techniques allows to achieve trust among strangers without the need of a *"trusted"* mediator.

2.1.1 Transactions

In the previous chapter it has been explained how a blockchain works from a high level point of view, but without introducing more in detail the mechanism that allows to create and resolve transactions, it would be very hard to actually understand the previous concepts. Although the project on which the thesis will focus doesn't involve the direct use of cryptocurrency exchange, it is necessary to analyze this process for a correct understanding of what happens under the hood of applications blockchain based. In the next paragraphs, we will take Bitcoin as an example to explore transactions work-flow.

Before diving into an explanation of the steps to follow to make a payment using the Bitcoin blockchain, we must explain two more concepts:

- **Asymmetric Cryptography.** It is a cryptography technique that involves the use of a pair of keys, one private and one public. Both of them are strictly correlated since the data encrypted with one of the keys can only be decrypted

with the other one.

The private key is the one that the user is supposed to keep secret, while the public key is the one shared with anyone the user wants to communicate with. If user `A` wants to send a message to user `B`, he has to cipher the message with his private key. `B` is now able to decipher the message using `A`'s public key previously shared. When a message is sent in this direction, the mechanism does not offer the **secrecy** property since everyone potentially knows the public key and so everyone can read the message. However, the mechanism guarantees **integrity** (if the encrypted message is altered, it cannot be deciphered) and **authentication** (anyone can check the identity of the author) properties, ensuring to `B` that the message has been written by `A` because only the public key related to the `A`'s private one can decipher messages encrypted with the latter. This property is called, instead, **non-repudiation**.

We can now analyze a message exchanged in the opposite direction. If `B` sends a message to `A`, the first has to cipher it with the public key of `A`. In this case no one can decrypt the message that `B` sent because the only key capable of extracting the message is the private key of `A`, known by him only. The secrecy of the message is now achieved, instead non-repudiation and authentication are lost because `A` cannot know who is the author of the message due to the shared public key. Message integrity is still preserved.

- **Digital Signature.** It is an application of asymmetric cryptography that aims to ensure integrity, authentication and non-repudiation properties. To create a digital signature, the author uses his private key to encrypt the digest (a summary of a fixed length, i.e. 256 bit, of the considered data, to avoid doubling the size of the file) of a certain document he wants to sign. Then he sends the plain document and attach to it the digital signature and a certificate that contains the author's public key. The receiver can check the mentioned properties as follows (`Figure 2.1`):

  - **Authentication:** he decrypts the digital signature using the public key of the author. He obtains the digest of the document.
  - **Integrity:** he re-computes the digest of the plain document and compares it with the one extracted before. If he finds a match, the document hasn't been altered.
  - **Non-repudiation:** if the previous steps went fine, the author cannot deny he sent the message.

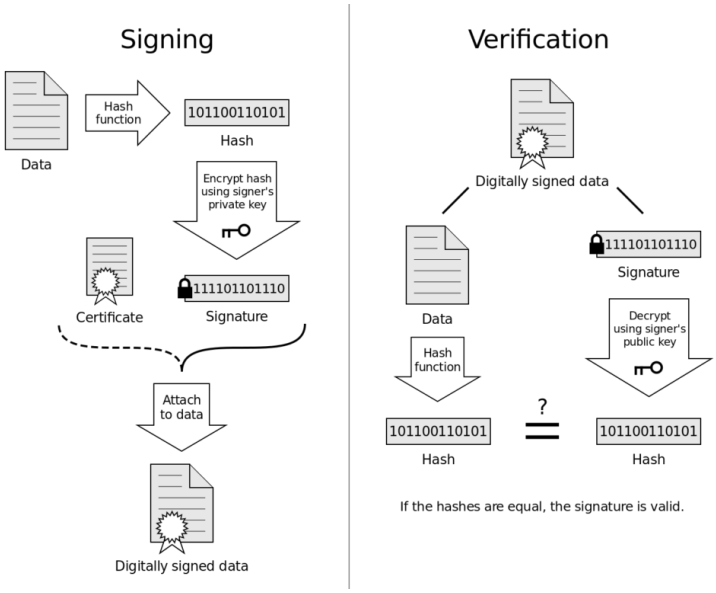

Figure 2.1: Digital signature flow chart[1].

At this point, we suppose that two users, *Alice* and *Bob*, want to share a few bitcoins. We also suppose that both *Alice* and *Bob* have a crypto wallet. In short, a crypto wallet is an application, runnable on computers, smartphones or dedicated hardwares, that allows users to store and retrieve cryptocurrencies such as Bitcoin. Finally we say that *Alice* wants to pay *Bob* using her cryptocurrency. In order to make the payment successfully, these steps must be followed:

1. *Bob* has to create a new address for his wallet in which he can receive the coins sent by *Alice*. An address is a sequence of bits (160 for Bitcoin) generated starting from a pair of private and public keys. To calculate the address, an **Elliptic Curve Cryptosystems** must be chosen. Currently Bitcoin uses **Secp256k1**[2] with the **ECDSA**[3] (*Elliptic Curve Digital Signature Algorithm*). *Bob* can now share the new address with *Alice* so that she can do the payment.

2. *Alice* uses one of her addresses with a positive balance as the source of the transaction and *Bob*'s previously shared address as destination. Finally, *Alice* signs the transaction with her private key.

[1]Source: www.crs4.it

[2]Secp256k1 refers to the parameters of the elliptic curve used in Bitcoin's public-key cryptography, and is defined in Standards for Efficient Cryptography (SEC)

[3]ECDSA is a cryptographic algorithm used by Bitcoin to ensure that funds can only be spent by their rightful owners

3. Each node of the network that has visibility of the transaction, can verify it through a balance check of the source address and a verification of the digital signature authenticity. Then, the mining process starts involving each node of the network.

4. Miners begin to collect a certain amount of transactions and group them into a block to validate, and the competition begins.

5. Once the PoW has been found, and verified by the others, the block is attached to the blockchain. The winner receives now the reward for its effort as new minted bitcoins.

2.2 Hashcash in Bitcoin mining

We just explained what are the steps that are executed in order to do a bitcoin exchange using the blockchain. But how is actually computed the Bitcoin's PoW?

Satoshi Nakamoto mentioned that a certain algorithm, called Hashcash, was taken as an inspiration to create the entire consensus protocol of Bitcoin. Hashcash refers to a proof of work system that was created by Adam Back in 1997 and it was initially meant to limit spamming and DDoS attacks.

The hashcash algorithm is relatively simple to understand. The idea builds on a security property of cryptographic hashes, that they are designed to be hard to invert (so-called pre-image resistant property). You can compute y from x as $y = H(x)$ cheaply, but it's very hard to find x given only y. A full hash inversion has a known computationally unfeasible brute-force (trial and error approach) running time, being $O(2^k)$ where k is the hash size (e.g SHA256, $k = 256$). If a pre-image was found, anyone could very efficiently verify it by computing one single hash: this implies a huge asymmetry in full pre-image mining (computationally unfeasible) against verification (a single hash invocation).

We then define a second pre-image as: given one pre-image x and y so that $y = H(x)$, the task is to find another pre-image x' (the second pre-image) of hash y so that $y = H(x')$. It is important not to confuse with a birthday collision which is to find two values x, x' so that $H(x) = H(x')$: this can be done in much lower work $O(\sqrt{2^k}) = O(2^{k/2})$ because you can proceed by computing many $H(x)$ values and storing them until you find a matching pair. It takes a lot of memory, but there are memory-time trade-offs.

Version 0 of hashcash protocol (1997) used a **partial second pre-image**, however the later version 1 (2002) uses **partial pre-images of a fairly chosen string, rather than digits of π or something arbitrary**: 0^k (i.e all 0 bits) is used for convenience. In this case, the work is to find x such that $H(x) = 0$. To make it easier, the definition of a partial pre-image is to find x such that $H(x)/2^{n-k} = 0$ where "/" is the integer quotient from division, n is the size of the hash output (n=256-bits for SHA256) and k is the work factor, which is the number of the firsts bits of the hash output equal to 0. For example, if $k = 20$ and $n = 256$, the task is to find y such that $H(x) = y$, with y starting with 20 zero bits while the remaining 236 bits can be anything ($H(x)/2^{236} = 0$). It is actually the hash output (the digest) that partially matches the chosen condition (0^k), not the pre-image. In fact, it could be more accurately called a pre-image with a partial output match[12].

Bitcoin's PoW is very similar to the just described algorithm, but there are few differences: once a miner has chosen the set of transactions to put into a block, he computes the hash of the block by using twice the SHA256 hash function on the body of the block, while changing at each iteration the so called `Nonce` until the output of the hash match the specified target. The target is the threshold number that a block digest must be below, in order to add the block to the blockchain. The target concept replaces the hashcash work factor. The `Nonce`, instead, is basically a random value that is concatenated to the body of the block; once it is changed, also the computed hash output will change and this procedure is done until the correct `Nonce` is found. To summarize, Bitcoin's PoW consists of finding a `Nonce`, using a brute-force approach, so that the digest of the body of the block concatenated to it, is smaller than the target.
An example could be needed to make the process simpler to understand: we suppose that our target is 2^{240} and we must find an output of the SHA-256 hash function that is smaller than the target. We vary the string (that in this example represents the block) by adding an integer value to the end (the `Nonce`) incrementing it at each iteration. Then, interpreting the hash result as a long integer and checking whether it's smaller than 2^{240}. To find a match for `"Hello, world!"` 4251 tries are needed[13].

```
"Hello, world!0" =>
1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64
= 2^252.253458683

"Hello, world!1" =>
e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8
= 2^255.868431117
```

```
"Hello, world!2" =>
    ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7
= 2^255.444730341

[...]

"Hello, world!4248" =>
6e110d98b388e77e9c6f042ac6b497cec46660deef75a55ebc7cfdf65cc0b965
= 2^254.782233115

"Hello, world!4249" =>
c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6
= 2^255.585082774

"Hello, world!4250" =>
0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9
= 2^239.61238653
```

When the consensus is reached and the block is attached to the blockchain, the hash of the header of the block is computed, and this will be the fingerprint of the entire block that, from now on, will represent it uniquely and will be used as a reference to it by the next block, as shown in `Figure 2.2`.

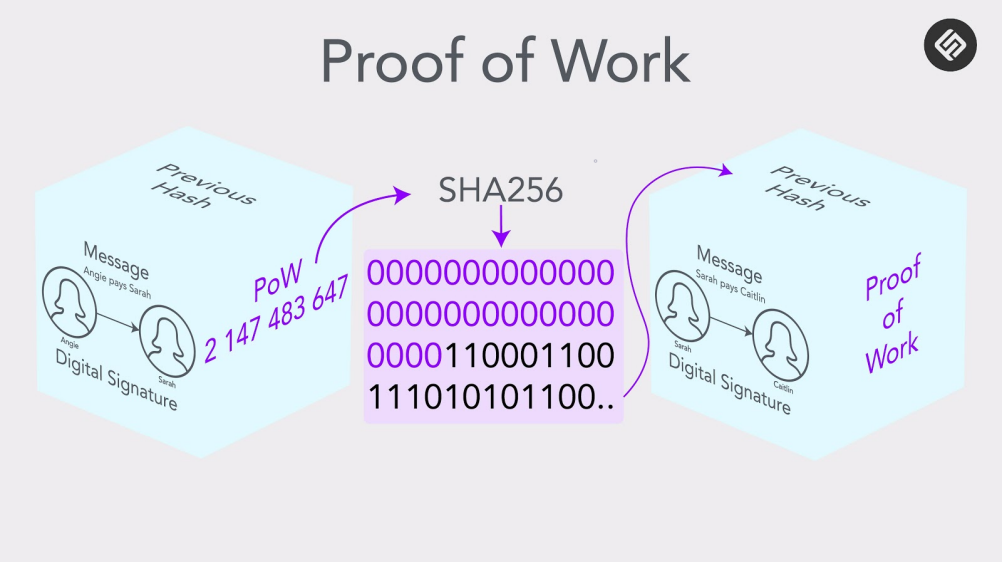

Figure 2.2: Bitcoin's proof of work[4]

[4]Source: dev.to

2.3 Forks on the Bitcoin's Blockchain

When a distributed network, that relies on a consensus protocol PoW based, includes a huge amount of nodes, the probability that two of them find the PoW in a small time interval increases. We refer to this phenomenon as a **fork**, and when it happens, a fair procedure, allowing to choose one of the blocks over the others, needs to be found because of the intrinsic architecture of the blockchain. Once a block is chosen the other one becomes invalid and sooner or later will be discharged.

But how should this procedure act? When two PoWs are computed and sent to the other nodes in a really small time interval, it's impossible to accept one over the other just relying on the arrival time because the asynchrony of the network hinders validators to recognize who the winner is: this leads to the creation of a fork.

When the network has to deal with a fork, each of the two blocks are temporarily considered valid as a possible block to attach to the blockchain. At this point, miners can decide which of the two branches of the fork they will keep mining on.

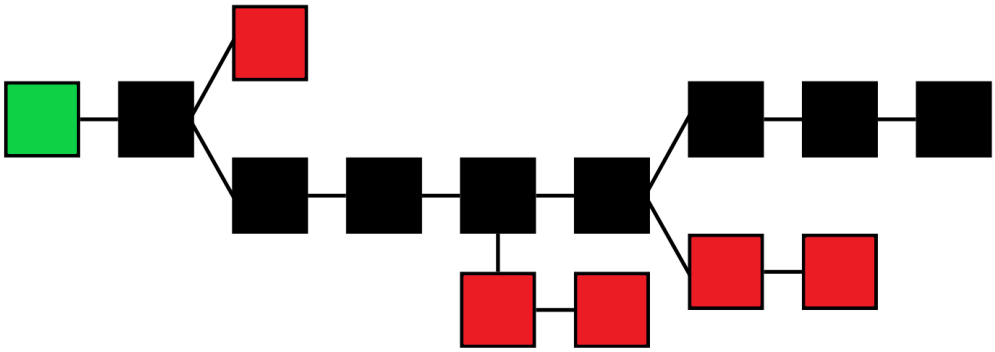

Figure 2.3: Example of a blockchain fork.

Once a new block is added to one of the two branches, miners tend to move to the longest chain, because it has a greater probability to become the final one. Another motivation of this behaviour is that transactions within the discharged blocks will not be validated, so miners will get no earnings for their efforts. In other words, Bitcoin's blockchain follows the rule `"the longest chain wins"`. Thanks to this mechanism, forks are usually resolved quickly: the longest fork in Bitcoin's history contains just six blocks and that is why a block that receives at least six confirmations is considered secure. The `Figure 2.3` shows a representation of forks of a blockchain, where the green block is the genesis block, the black blocks compose the main chain and the red blocks are the discharged ones after a fork.

2.3.1 Soft fork and Hard fork

We just described forks caused accidentally by the behaviour that PoW protocol leads to. However, there are other reasons that could lead to the creation of a fork. For example, when some protocol's modifications are proposed, miners can accept or reject these changes.

If a global consensus is reached, no fork will be created and each member of the network upgrades its version of the software to make the system keep working correctly with the new rules. Instead, if the consensus never comes, two factions will be created: the innovators that are in favour of the new rules, and the conservatives that want to keep the old ones.
At this point, innovators can decide to give up or to continue on their own way. If they continue, they have two ways to proceed:

1. **Soft fork:** guaranteeing backwards compatibility with the old software, so that the conservatives can still work together with the innovators. In this case there is not a real fork, because the main chain is still one.

2. **Hard fork:** upgrade the software with changes that make it incompatible with the old one. In this case two blockchains will grow in parallel, still sharing the same blocks until the fork. Innovators and conservatives are now split and will keep mining in different blockchains, but also users and developers usually tend to choose just one of those.

In Bitcoin's history several hard forks have occurred: the first one is called Bitcoin Cash.

2.4 Blocks Validation Rate

The greatest problem of Bitcoin comes from the need to keep the number of the attached blocks in the blockchain, in a certain time interval, below a very small threshold. This threshold is set to allow the validation of a single block every ten minutes. This is required because if the validation rate increases too much, the forks problem could become more and more complex to solve, extending the time needed to be sure that a certain block will not be discharged during the resolution of a fork.

Because of the continuous growth of the Bitcoin's network and consequently of the increase of the validation rate, the PoW target is always adjusted (decreased generally) to maintain the average time of a block insertion around ten minutes.

We define difficulty, the ration between the initial target and the current one. The target is updated every 2016 blocks validations (around two weeks):

- If the average of the validation time computed on 2016 blocks is less than ten minutes for each block, the target decreases.
- On the contrary, the target increases.

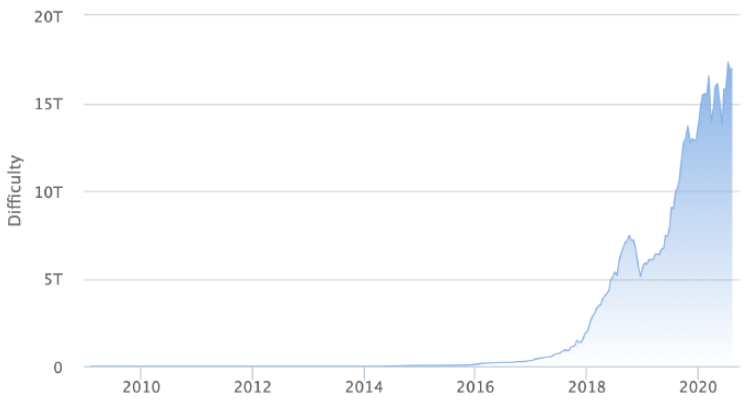

Figure 2.4: Bitcoin's difficulty trend[5].

Considering that each block has a maximum size of 1MB, that it contains an average of 2000/3000 transactions and that a block is appended to the blockchain every 10 minutes, Bitcoin's network is capable of validating at most seven transactions per second, but the average is just three or four.

This low validation rate causes the scalability problem previously mentioned that hinders the entire project to become a real alternative for the global financial system. If we do a comparison between Bitcoin and whatever digital payments big player such as Visa, Mastercard, or Paypal, we can notice how the first one has a validation rate of several orders of magnitude smaller, as we can see in `Figure 2.5`.

Another consequence of the slow payment speed, is that the network is not capable of satisfying all the incoming transactions. For this reason a transaction becomes more important as the fees that users are willing to pay are higher. In fact,

[5]Source: slides for the course *Blockchain e Criptoeconomia* of Politecnico di Torino - Danilo Bazzanella

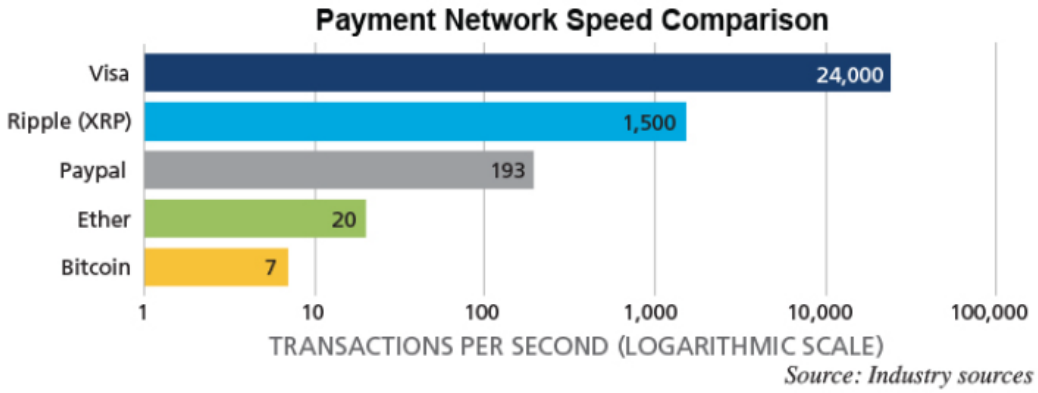

Figure 2.5: Validation rate comparison[6].

miners are pushed to first validate the transactions that promise higher earnings, making the network inappropriate for microtransactions.

2.5 Blocks Structure

In the previous sections, we described what a block is, without going in detail. The set of transactions within a block are organized in order to meet the blockchain specifications. The block includes, indeed, some metadata for this purpose. We now explore block structure in the specific case of Bitcoin, but similar solutions are used in other blockchains.

Blocks are made of two parts: the **header** and the **body**; the first contains metadata needed for the correct management of blocks inside the blockchain, while the second contains the set of transactions. In `Table 2.1` are reported all the information stored in the header of a block.

At this point we are able to recognize almost every field and its meaning, except for the `Version` and the `hashMerkleRoot` fields that are respectively the version number of the protocol used by the blockchain and the SHA-256 hash output of the set of transactions stored in the body.

[6]Source: slides for the course *Blockchain e Criptoeconomia* of Politecnico di Torino - Danilo Bazzanella

Field	Purpose	Size (Bytes)
`Version`	Block version number	4
`hashPrevBlock`	SHA-256 of the previous block header	32
`hashMerkleRoot`	SHA-256 based on all of the transactions in the block	32
`Time`	Current *block timestamp* as seconds since 1970-01-01T00:00 UTC	4
`Bits`	Current *target* in compact format	4
`Nonce`	32-bit number (starts at 0)	4

Table 2.1: Block's Header of Bitcoin

During the blockchain introduction, we said that transactions in a block are organized in a data structure called **Merkle Tree**. This data structure permits to efficiently verify if a transaction belongs to a certain block using only a small subset of the entire transaction set. In this way data overhead is avoided, and less data needs to be exchanged in the network.

`Figure 2.6` represents a Merkle Tree: we can define it as a binary tree where each node is the combined hash of the two child nodes, and the leaves are the plain transactions. The root of a Merkle Tree is called **merkle root**. This is the resulting hash computed starting from the leaves that are combined in couples to compute the hash of their parent node, and the process continues until it reaches the root. But what such a data structure is convenient for?

To understand the advantage of using a Merkle Tree, we make an example: we suppose that we need to verify if the green transaction (H_K) belongs to the block. To do so, we have to compute the hashes at each level of the tree and check if the hash of the merkle root matches the computed one. In the same figure we notice how the data needed to make all the computations are the blue nodes: so instead of using the entire transaction set (16 transactions) we just need 3 hashes and the near transaction (H_L) corresponding to the blue nodes to recompute the hash of the Merkle Root. Considering that a single transaction is in size greater than 256 bit (the hash size), and that only $h - 1$ (where h is the height of the tree) hashes plus a plain transaction is needed, it's easy to understand that the amount of the exchanged data is drastically reduced.

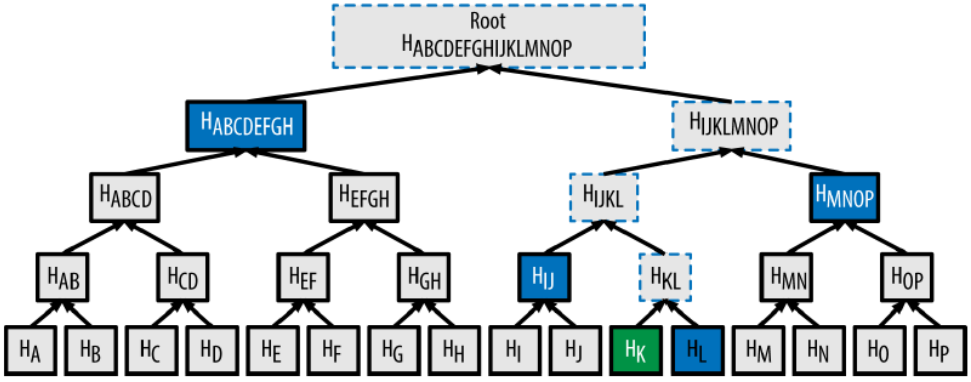

Figure 2.6: Merkle Tree representation[7].

2.6 Weaknesses and Limitations of Bitcoin

One of the most important properties, that a blockchain must have, is the property of being secure. Despite being designed to be so, blockchains may be subject to some vulnerabilities that can affect the health of the system making it unstable, unusable or simply less secure. Even if these vulnerabilities have little probabilities to cause damage to the system, it is important to know how they work and how to prevent them, because they could be a problem in the future, when new technologies could be exploited to make cyber-attacks more effective.

An overview of the possible attacks are here reported, considering the Bitcoin network. Still, almost all of them are common to every blockchain implementation.

- **Sybil Attack.** It is a threat for online systems where a single user tries to take control over the network by creating a multitude of different accounts, that in this case are the nodes of the network.

  Attackers that use this kind of attack could reach the majority of the network causing lots of problems [14]:

  – They can refuse to receive and transmit every transaction, effectively disconnecting users from the network, and making them unable to use that service.

[7]Source: slides for the course *Blockchain e Criptoeconomia* of Politecnico di Torino - Danilo Bazzanella

  - They can relay only blocks that they create, putting users on a separate network and then also leaving them open to double-spending attacks.
  - If you rely on a transaction with 0 confirmations, they can just filter out certain transactions to execute double-spending attacks.

  At the moment, mechanisms that guarantee complete security against these attacks do not exist, but blockchains and, more specifically, consensus protocols make sybil attacks unfeasible.

  For example, in the Bitcoin network, the ability to forge a block must rely on the difficulty imposed by the PoW target. This means that for being capable of computing the PoW, the node requires a certain amount of computational power that is far from being negligible. This would bring attackers to use their financial resources to buy the necessary hardwares to execute the attack. If the attackers reach a computational power greater than 50% of the whole network's power, they can basically control the entire chain forcing consensus for fraudulent or double-spend transactions or simply excluding honest ones. This situation is usually labelled as **51% attack**. That is why sybil attacks (and 51% attacks consequently) are considered quite unfeasible as long as the network includes a great number of nodes.

- **Double-spend attack.** We mentioned double-spending attacks as one of the problems that could ruin a whole blockchain based system. Double-spending is the risk that a digital currency can be spent twice. It is a potential problem unique to digital currencies because digital information can be reproduced relatively easily by attackers that have great knowledge of the interested network.

  Reusing the hypothesis that an attacker has access to more than half of the whole network computing power, he has to follow the following step in order to do a double-spend attack (representation in `Figure 2.7`):

  1. First, the attacker has to make two payments with the same bitcoins; one to an online vendor and the other to another address owned by himself.
  2. Initially he will only send the payment to the vendor.
  3. Once the transaction is included in a block, the vendor can check it and send the products ordered by the attacker.
  4. When the attacker is sure that its order was processed and products sent, he will use all of its computing power to create a secondary branch that sooner or later will be longer than the main one. In this branch the attacker will include the transaction to its secondary address and not the one to the vendor, which will be discarded.

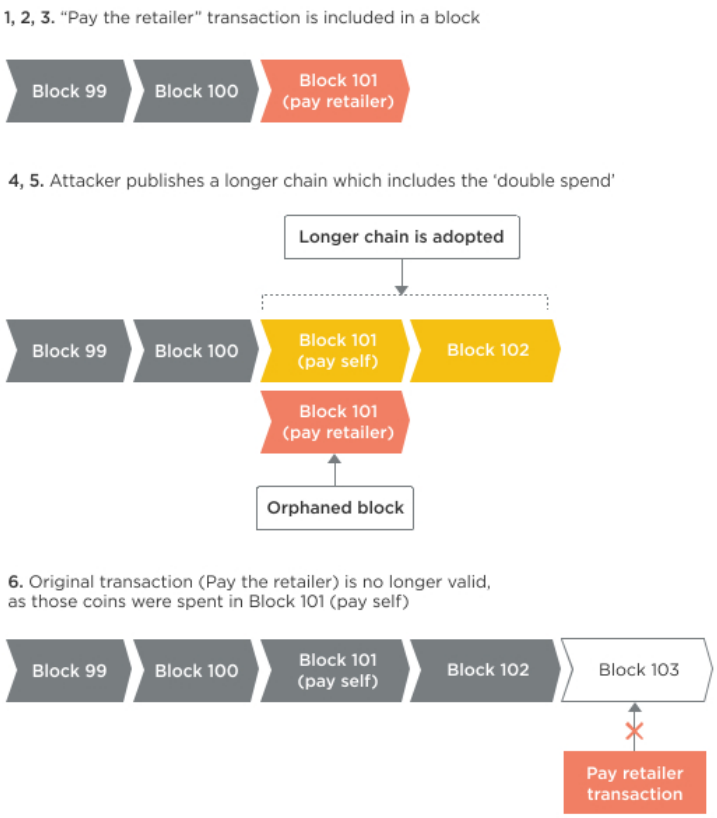

Figure 2.7: Representation of a double-spending attack workflow[8].

5. Once the secondary branch becomes longer than the main one, it will be considered the correct branch and each node will keep adding other blocks to it.
6. The original payment is considered invalid even from the honest nodes because the bitcoins used for the transaction are already spent.

Other noteworthy problems exist in the context of the blockchain of Bitcoin but they are even less probable to happen than the previous ones, or they cause minor damage to the system[14]:

- **Possibility to attach illegal content to the transactions.** In fact, since arbitrary data can be included in Bitcoin transactions, and full Bitcoin nodes

[8]Source: medium.com

must normally have a copy of all unspent transactions, this could cause legal problems. However, local node policy generally doesn't permit arbitrary data.

- **Breaking the cryptography.** SHA-256 and ECDSA are considered very strong currently, but they might be broken in the far future. If that happens, Bitcoin can shift to a stronger algorithm.
- **Tracing a coin's history.** It can be used to connect identities to addresses.

We then report again the problems related mainly to the PoW consensus protocol that implies a huge energy consumption for the mining process and a poor scalable system.

At this point, there are a couple questions that it is legitimate to think as potential problems[14]:

- ***Is it possible that two people with different private keys generate the same address?*** Collisions are highly unlikely: keys are 256 bit in length and they are hashed in a 160 bit address ($2^{160^{th}}$ power). Divide it by the world population and we have about 2.15×10^{38} addresses per capita.
- ***What if everyone calculates PoW at the same rate?*** If everyone began with identical blocks, they starts their nonce at 1 and increment it at each iteration, the fastest machine would always win. However, each miner builds its own block in a random and independent manner. So everyone begins with slightly different blocks and everyone truly has a random chance of winning (even if still proportional to the computational power).

As we can easily notice, Bitcoin is overall a secure and reliable platform that has been designed from the beginning with these problems in mind, and that is why lots of people consider blockchains and cryptocurrencies as possible candidates for the finance of the future.

Chapter 3

Blockchain 2.0: Tools and Use Cases

So far, when the word *"blockchain"* has been mentioned, we have always referred to Bitcoin, but blockchain applications are almost unlimited and go beyond the concepts of Bitcoin and cryptocurrency in general.

The blockchain technology and its capability to create more transparency and fairness while also helping businesses in saving financial and time resources, is impacting a wide range of sectors, many of which are far away from the financial one. Today it is a real **game-changer** for multiple industries including health-care, education, real estate, supply chains and logistics, or even **circular economy** scenarios which are the ones we will be focusing on before introducing the **BioEnPro4To project**.

Blockchain technology is evolving constantly, revealing itself to be more flexible and versatile than might have been expected a few years ago. The reason for the exponential growth of its progress, might be found in the advent of another big player in the game. *Ethereum*, officially released the 30^{th} July of 2015, lays the foundation for a modern concept of Blockchain introducing new tools such as ***smart contracts*** and a protocol to implement tokenization on top of its main layer. Ethereum starts the **blockchain 2.0 era**[1].

[1]Instead of viewing the blockchain just as financial decentralisation, blockchain 2.0 expands the scope of the technology enabling decentralization of markets in general.

3.1 Main Features of Blockchain 2.0

3.1.1 Smart Contracts

The concept of *smart contract* is somehow attributable to Nick Szabo that in one of his works, in the 90's, he is the first to use the term, stating that *"A smart contract is a computerized transaction protocol that executes the terms of a contract"*[15]. So a smart contract is nothing but a software to automate the execution of some predefined contractual obligations. They are written using a particular programming language that runs, generally, on a trusted computer network that uses a specific protocol. More in detail they can be classified under the label of **event-driven software**, representing contractual obligations in a digital form. Using these definitions, a smart contract is non necessarily related to blockchains and it is trusted and unbiased by design: once the rules of the contract have been defined, the program will always produce the same output by entering the same inputs.

In the current industrial practice, a smart contract is a software that is executed by the nodes of a blockchain network and so it would benefit from the same advantages of a DLT, in particular decentralization, distributed execution, and strong security. Every time a smart contract produces an output, it results in a change of blockchain state that is represented as a transaction the miners/validators have to validate respecting the rules defined by the consensus protocol. In this meaning, they are not just contracts but more generally softwares whose executions and outputs are guaranteed to be intact and trusted by the underlying technology properties. The naming convention for these distributed applications (also known as **dApps**) has been chosen by Ethereum that first introduced this tool in the sector.

The main factor to consider while writing a smart contract is that, as any other information that is published on a blockchain, it will be immutable over time. So it is easy to understand that when a smart contract is released, it must guarantee its reliability by running a bug-free code and using data sources, trusted by each involved party, to trigger events, and so producing the expected outputs reflecting the predefined rules.

A practical example of a smart contract could be written to represents this agreement:

- If company ***C*** manages to raise funds totaling $1.000.000, investors ***I*** will earn a percentage of profits generated by ***C*** weighted by their initial investment amount.

- Otherwise, if the invested funds don't reach the target by the end of date ***T***, each investor will be refunded.

In this simple contract, we can find two events with their respective consequences:

Event	**Consequence**
Invested funds reach the target ($1.000.000)	Investors will earn some profits
Invested funds don't reach the target by the end of a certain amount of time	Investors will be refunded

In this example the programmed smart contract would act as a neutral intermediary and the involved parties, ***C*** and ***I***, would benefit from all the guarantees that an external intermediary could offer but without any negative aspects to accept, such as commission to pay for the provided service. Once one of the two events happens, the smart contract will trigger the corresponding function to reflect the corresponding effect. Before an event happens, the smart contract could change its state several times.

For example the initial state of the smart contract could be like the one below:

```
Investor, Fund = [ ]
Total Funds = $0
Target = $1.000.000
Expiration time = 31/12/2021
Remaining time = 6 months left (External data source)
```

The state of the smart contract traces every relevant information for the correct conclusion of the contract. In this case it stores: for each investor how much money he put in the contract, the total amount of received funds, the goal to reach and the date of the expiration of the contract in case the target will not be reached. This data can be stored statically in the state, so for each new input (new funds from an investor), the state of the contract is updated and a new transaction is issued in the underlying blockchain. The information of the remaining time is instead computed by the smart contract by using an external data source which should be, in this case, a time source that investors trust once they decide to send funds to the contract.

An intermediate state could appear instead like this:

```
Investor, Fund = [ (I1, $100.000), (I2, $300.000) ]
Total Funds = $400.000
Target = $1.000.000
Expiration time = 31/12/2021
Remaining time = 3 months left (External data source)
```

We can see that two investors have sent funds to the contract and a total of $400.000/$1.000.000 has been collected. Internally it manages the collected funds by owning an address on that particular blockchain, and thanks to this it can move the funds independently once one of the two events happens by sending the total funds to the address of the company ***C*** and guaranteeing earnings for the investors in the first case, or just by refunding the investors ***I*** sending back to their address the money.

3.1.2 Tokenization

Tokenization is the process to represent some assets, both physical or digital, into a token that can be moved, stored and recorded on a blockchain. To say it in other words, tokenization converts the value of an object, even if it would be normally impossible to subdivide, into one or more tokens that can be manipulated for some reason on a blockchain. By using tokenization, each item would become digitally tradable in a more secure and faster manner, overcoming the limitations of the traditional paper markets. Moreover, trading assets does not require an intermediary strengthening once more the concept of decentralization.

To simply understand how a crypto-token works, we can think of how real tokens work. For example when we go to the cinema, before entering the room where the movie is screened, we must present the ticket to the operator and before that, we have to buy it at the ticket office. So in this case, the ticket represents our token and by buying the token we are buying the permission to enter the room and to watch the movie. But why is the ticket needed and why do we pay for a piece of paper which has no real value? The answer for the first question is about necessities: introducing a *two-steps access* for the movie room instead of one, basically solves "logistic" problems. In fact, if the *two-step access* is not used, there would be the need for a ticket office near each room of the cinema, while using the tickets method, the ticket office can be just at entrance while operators can checks the validity of the tickets near the rooms. The answer for the second question is instead so obvious, that maybe just a few have ever thought about it: we agree to buy a valueless piece of paper, just because we are sure that there is someone that recognizes the value that exist behind the ticket. The same concept is applicable

to any token representing any other asset.

Although cryptocurrency coins and tokens might appear as synonyms at first glance, they are far from being the same thing. The largest difference between them is that, a cryptocurrency coin is native to its very own blockchain, while a token does not require its own blockchain but it can be easily created by a template compatible with a certain underlying blockchain. Ethereum is today the blockchain with the largest token ecosystem, thanks to its ERC-20 and ERC-721 standards[2].

Tokenization of assets has no limits but each corresponding token can be grouped in one of three different categories:

1. **Intangible tokens.** In this category we find assets that exist only due to the operation of law and no physical objects exist (copyrights, patents, etc)[16].

2. **Fungible tokens.** They are assets that can be replaced by another identical item. They represent the object itself and so its value. Examples could be gold, dollars etc. These kinds of items are easy to be tokenized because they can be divided into smaller units or grouped into bigger ones. So a token can stand, for example, both for a fraction of a gold coin or more units of them
   To tokenize these assets it is required an abstraction layer: a set of tokens must be always assured by the corresponding asset. For example for the case of USDTs tokens, each one of them is pegged to a real assured dollar. The advantage is now that tokens representing dollars can be easily exchanged in the Ethereum blockchain.

3. **Non-fungible tokens (NFTs).** These tokens represent instead non-fungible goods which are those which can't be broken down into smaller pieces in the real world. Tokenization overcomes this limitation allowing to break down non-fungible assets into digital shares which can now be traded fully or in a limited fashion. Collectibles items or real estate are two of the best examples for this category.
   Taking for example a painting from a famous painter, we can easily affirm that there is only one authentic painting and that posters that replicate the image of the painting are not the same thing and have not the same value.

   To tokenize an item like this, a digital signature is introduced in order to guarantee the authenticity of the painting. From now on, this digital NFT

[2]The ERC-20 and the ERC-721 introduce a standard for Fungible and Non-fungible tokens respectively.

represents the painting as a digital twin and it is unique as the painting itself. But the token can now be broken down into sub-tokens also digitally signed, which can be sold like shares of the original painting to the public.

3.2 A Use Case Analysis: Circular Bioeconomy for Waste Management

In recent years, global sustainability challenges, such as world population growth and over-consumption of non-renewable resources, are pushing societies to establish and follow circular and regenerative approaches. These approaches are represented in particular by two economic models that can be wisely interlaced to enable the positive aspects of both.

Before continuing, it's necessary a brief digression about the concepts of bioeconomy and circular economy:

- **Bioeconomy.** As reported on Wikipedia, bioeconomy *"refers to economic activity involving the use of biotechnology and biomass in the production of goods, services, or energy"*[17]. The concept derives from the need to adopt an economic model that is **ecologically** and **socially** sustainable. Whatever economic process, focused on the production of material goods, will decrease the energy availability to produce other goods in the future. Georgescu-Roegen is the author of this theory, and he states also that once raw materials are spread in the environment, they can be reused only to a lesser extent and with a greater waste of energy. That's why bioeconomy is today an active field of research.

- **Circular economy.** This economic system is designed to be completely self-sustaining: it is a production and consumption model that carefully focuses on reducing the wastes of natural resources, and contemporary enhancing the reuse and the recycling of the existing products as long as possible. Whenever a product reaches the end of its lifecycle, materials of which it is made of, will be re-injected in the market, if possible. These materials will be re-used to assemble other products, restarting the entire loop.
  The promotion of circular economy relies mainly on two cornerstones:

  1. **Reduction of the amount of wastes.** This would be feasible through some preventive measures to apply during the designing and production processes of the product
  2. **Diffusion of aware recycling behaviours.**

3.2.1 Issues Overview

Bioeconomy and circular economy can help to alleviate environmental problems caused by the lack of modern and smart waste management systems, which do not provide operational transparency, traceability, security and trusted data provenance features. In fact, today's technologies leveraged for waste management, strictly relies on manual and centralized silos, and this makes them vulnerable and the single point of failure of the entire system.

As introduced before, a circular economy system employs a reuse, repair, refurbish, and recycling model to create a closed-loop system where waste materials and energy become inputs for other processes. The circular economy is a good match with the bioeconomy model, which promotes the enhanced use of organic waste and the recycling of biological resources. The conversion of renewable bio-resources into value-added products is part of a circular system that can make businesses more economically viable and sustainable in the long term.
In EU countries, the *Waste Framework Directive*[3] demands a more virtuous use of wastes and sets specific goals, such as the target of re-use and the recycling of municipal waste to a minimum of 65% by weight by 2035 and the goal to reduce the landfilling of municipal waste to 10% by 2030.

To reach these goals, the transition plan must consider integrating economic and environmental directives as well as stimulating social awareness, leading to more conscious behaviours. Organic resources such as food wastes, biodegradable and compostable bags and packaging, can be transformed into organic fertilizer or in organic gas that can be used as fuel. The process of transforming biomasses exploits several techniques, such as anaerobic digestion, pyrolysis, torrefaction, fermentation. However, currently they are not optimally utilized. Waste resources are complex materials that are strongly varying in terms of composition, quantity and quality. Unknown quality of waste resources might hinder their use as fertilizer, since they can contain environmental pollutants, pathogenic microorganisms and microplastics. This is why it is essential to ensure the quality of the inputs to get high value products.

[3]It sets the basic concepts and definitions related to waste management, including definitions of waste, recycling and recovery.

3.2.2 Digital Ledgers Technologies and Circular Bioeconomy

Due to their nature, DLTs are today attractive tools to solve the main supply chain management challenges on circular economy use cases. A huge variety of different sectors, such as the fast-fashion one, are approaching a circular economy model in their businesses by including innovative frameworks or services, blockchain based, into their tech-stacks to enable transparency, traceability and security. Especially for the fast-fashion sector, it is possible to already find multiple examples of architectures and strategies that exploit very well the possibilities offered by different blockchains. On the other hand, this is not the case for the waste management sector, and that is why this thesis aims at giving a little contribution by proposing a PoC for a real use case.

Tracing and tracking features have been mentioned more than once, but why would they be so important for this purpose? All the amount of waste resources produced by a (smart) city should be sent to landfills, waste recycling facilities, composters and waste-to-energy generation plants. In this process, traceability can be very useful to verify the authenticity of data and ethical practices involved in the collection, processing, and shipment of waste. These kinds of features would assist in monitoring the current location and state of the waste during their treatment processes. It also offers an added value to the system since it assists in identifying, storing and managing detailed data about the activities and the outcomes of waste management operations[18]. Moreover traceability assures that each operation made during the different stages of the waste collection, have been done in compliance with the waste handling guidelines to protect the environment.

Finally a well designed system supported by robust infrastractures would enable users to efficiently track the end of life of the city waste, encouraging them to correct behaviours, for example, through a rewards system.

Chapter 4

Suitability Assessment of DLT Frameworks

Due to the huge number of available frameworks, it is important to establish the set of parameters to evaluate and pick the most suitable DLT for the considered use case.

BioEnPro4To is a project in the context of a circular bioeconomy model, that needs to be supported by a great quantity of sensors to efficiently monitoring waste treatment processes. This means that a massive amount of Internet-of-Things (IoT) devices is required. Because of this, to correctly and optimally exploits these devices and the related architecture of the infrastructure, we can identify three main factors to analyze in order to pick the most suitable distributed ledger platform:

- **Scalability.** Due to the nature of IoT scenarios, it is reasonable to think that a DLT should be able to manage a huge real-time data streaming. However, transactions on blockchains are often very slow due to the frequent adoption of the PoW as consensus protocol, as it is currently considered the most reliable one. This is the reason why the scalability is a weak point for a lot of blockchains.

- **Choice of the consensus protocol.** In addition to the massive adoption, the PoW is also well-known for its high-demanding computational power. Because of this, PoW may not be the best choice for an IoT context which is characterized almost entirely by low-power and low-energy devices. PoS may be a good alternative for this case, however, it has not fully tested yet.

- **Transaction fees.** The majority of blockchains are currently characterized by a certain cost that users have to pay, to miners or validators, in order to see

their transaction confirmed. This is due to the low scalability of the system, in fact, when the network is congested, miners and validators tend to favor users willing to pay higher fees to see their transaction confirmed. This is unfeasible for the analyzed scenario, where a large number of devices might have the necessity to send even hundreds of transactions per day.

4.1 DLT Frameworks Comparison

In collaboration with **Concept Reply**, we have considered a few DLTs implementation to compare, in order to explore and analyze benefits and disadvantages of each platform, in relation to the previously mentioned factors.

The comparison table below (`Table 4.1`) reports a summary of the most important features and characteristics of each chosen platform:

1. **Ethereum.** The blockchain with the **greatest ecosystem** right now. It guarantees a huge level of security and flexibility due to its solid smart contract infrastructure and token ecosystem.

2. **Hyperledger Fabric.** It is a **modular blockchain** framework oriented to developing blockchain-based products, solutions and applications using plug-and-play components that are aimed at a private enterprise use.

3. **IOTA.** Although it is still immature, IOTA is constantly growing day by day to provide a solid distributed platform that would offer the best tools for the **Internet of Things** and the **Machine-to-Machine** (M2M) economy, without compromising scalability and decentralization, at no cost for the end-user.

	Ethereum	**Hyperledger Fabric**	**IOTA**
Networks	Public (fees must be paid for transactions)	Private	Public (no transaction fees)
Vision	Create the greatest distributed platform with the largest ecosystem thanks to its solid smart contract programming language.	Create a modular distributed ledger platform to build ad hoc and private solutions for companies.	Provide an innovative distributed ledger platform to enable IoT and M2M economy, without compromising scalability and decentralization.
Consensus	PoW	Voting-based consensus protocol	Tangle + small PoW
Ledger Structure	Blockchain	Blockchain	Tangle
Future Improvements	Serenity Update *(Ethereum 2.0-2021/2022)*. It includes the transition to a PoS based consensus.	Minor improvements for the execution of smart contracts.	Coordicide Update *(IOTA 2.0-2021/2022)*. It includes improvements for consensus protocol, smart contracts and the removal of the Coordinator component to enable the true decentralization of the platform.

...	**Ethereum**	**Hyperledger Fabric**	**IOTA**
Advantages	**(i)** Flexibility offered by smart contracts that permit a huge variety of interaction with the ledger. **(ii)** It is the most mature and reliable platform of the three.	**(i)** Its consensus protocol offers a mechanism to reduce the computational and time cost for creating a transaction. **(ii)** It is a mature platform adopted by many companies in their business.	**(i)** It has been designed for IoT applications. This ensures low-end devices to be suitable to improve scalability and decentralization in the network. **(ii)** Feeless transactions
Disadvantages	**(i)** Every transaction on the blockchain has a cost, both economically and energetically. **(ii)** The current version is not IoT-compatible yet.	**(i)** The need to implement a private network leads to increase costs in terms of money and time for the development of a prototype. **(ii)** The architecture is not fully decentralized (Non-partition tolerant)	**(i)** The platform is still in a development phase, indeed many important features, just like smart contracts, are not fully available yet; **(ii)** After a certain amount of time, transactions that don't involve coin exchanges are deleted. To solve this problem, it is necessary to install a permanode called `Chronicle`.

Table 4.1: Features and characteristics comparison among DLT frameworks.

Considering the information reported in the comparison table, and without going into detail, we have drawn some conclusions:

- **Ethereum** is the most mature, reliable and versatile platform, however it does not suits very well for the purpose of the project: it is well known that the Ethereum is one of the most expensive blockchains in terms of

transaction fees and, although smart contracts could be an added value for future improvements, it is not, right now, the focus of the PoC. Finally it is not the best platform to interact within an IoT environment due to the low scalability.

- **Hyperledger Fabric**, on the other hand, fits better for a waste management system where low-power devices are a fundamental part of the architecture. However, its permissioned blockchain nature, makes Hyperledger Fabric inappropriate for a regional project as it is BioEnPro4To, in terms of maintaining costs, to give then free access to the citizens of the entire Piedmont. Moreover, aspects such as decentralization, transparency and resilience are weaker.

- **IOTA**, instead, offers interesting features for the case. It is natively designed for the world of the IoT, and its feeless nature allows easier integration of low-end devices. Moreover it provides a great variety of frameworks for developers that rely on the IOTA ledger, the Tangle, ensuring flexibility and great synergy among them to enable all the features required by the project. Finally it is a platform with great potential that might meet every future need of this complex circular bioeconomy scenario.

Chapter 5

History of IOTA

Digital revolution is characterized by the convergence of technologies: from cloud computing to edge computing, artificial intelligence, big data, IoT, and DLT. This is pushing communities to progressively switch from the physical to the digital world. In this context, the IoT adoption is increasingly becoming an integral feature of many modern systems often coupled with the use of DLT/blockchain. The fusion between IoT and DLT is an unprecedented paradigm shift that is expected to disrupt both current and future systems in various fields: in fact, blockchains have exactly what is needed to solve the vulnerabilities of IoT, in particular they solves security issues derived from the use of public trustless environment to connect each device to each other. Moreover the distributed and peer-to-peer nature of DLTs allows to easily handle the shortcomings of client/server models in Cloud-IoT solutions. However, as we know, we are at an early adoption stage because there are some significant challenges to overcome including scalability, data privacy, efficiency, storage, interoperability and more others. In addition to this, there is still no global consensus on what should be the best practices that specify how blockchain should be utilized in IoT[19].

This is where IOTA comes in. With this project the founders of IOTA paid particular attention to the sector of the IoT and M2M economy, enabling a huge variety of new possibilities through the implementation of a totally different DLT if compared to classic blockchains. If the project turns out to be successful, IOTA would allow micropayments that will let the IoT and DLT industry to rapidly grow, thanks to its fee-less model.

5.1 What is IOTA?

IOTA is an open-source project and cryptocurrency with the aim of being **faster**, **more efficient** and **more decentralized** than Bitcoin. The word *"IOTA"* also refers to the main protocol used by the ledger. What makes IOTA peculiar is the data structure used for its distributed ledger, called **The Tangle**. Differently from the most existing cryptocurrency that are based on blockchains, IOTA uses this particular ledger whose structure in math is known as **DAG** (*Directed Acyclic Graph*). The Tangle is a feeless and permissionless DLT used to track and process transactions. It has been designed to overcome the common blockchains limitations, such as low scalability or the need for prohibitive computational power deriving from the PoW consensus mechanisms, like for Bitcoin.

Thanks to the Tangle, IOTA is able to join the miners and users figures in one single entity. This concept removes the need for transaction fees and the mining process. All the IOTA coins (IOTAs) are already distributed, so when new transactions are validated, no more IOTAs are created. In fact there is no real financial incentive for the validators, contrary to Bitcoin and other PoW based blockchain: the IOTA project relies on the vision of a community driven free platform, designed to be more efficient as the number of nodes increases. Moreover, in the near future, even common devices such as smartphones or IoT ones would be able to validate transactions, so the incentive to join the network would be just making the system working better. Fortunately, despite this incentive could appear underestimated, the practice tells us differently, and the IOTA network keeps growing.

The lack of transaction fees enables microtransactions which are infeasible on blockchains such as Bitcoin or Ethereum.

IOTA is not just an application for cryptocurrency exchange, indeed it provides a series of tools that act as upper layers built on top of the Tangle in order to allow users develop actual applications for a great variety of use cases. From a general point of view, IOTA allows to develop applications with these features:

- **Authenticity and Integrity:** as it relies on a distributed ledger implementation, data are immutable and authentic by design.
- **Decentralisation:** IOTA is driven by a peer-to-peer network using a consensus protocol to guarantee the correct functioning of the system.
- **Confidentiality:** although the entire ledger is public and available to anyone, transactions are secured through digital signatures; moreover additional data can be made accessible only to certain selected users thanks to cryptography.

- **Microtransactions:** IOTA allows to send little amounts of the currency without any fee. This would open the doors to a huge number of new different approaches for many unsolved problems. When a node issues a new transaction on the network, it must approve two previous transactions that are awaiting for validations. This mechanism makes IOTA potentially infinitely scalable, because a greater number of new transactions implies a greater number of validated transactions, that would increase the speed and the security of the network.

5.2 The Tangle

The Tangle is a novel kind of distributed ledger architecture that is based, as we already mentioned, on a DAG. IOTA doesn't use the traditional design applied to most blockchain networks: instead of chaining blocks one after another, in the Tangle each block references two previous blocks. Because of this rule, the tangle evolves always along a specific direction that represents somehow a timeline that gives us a sort of chronological order for the transactions.

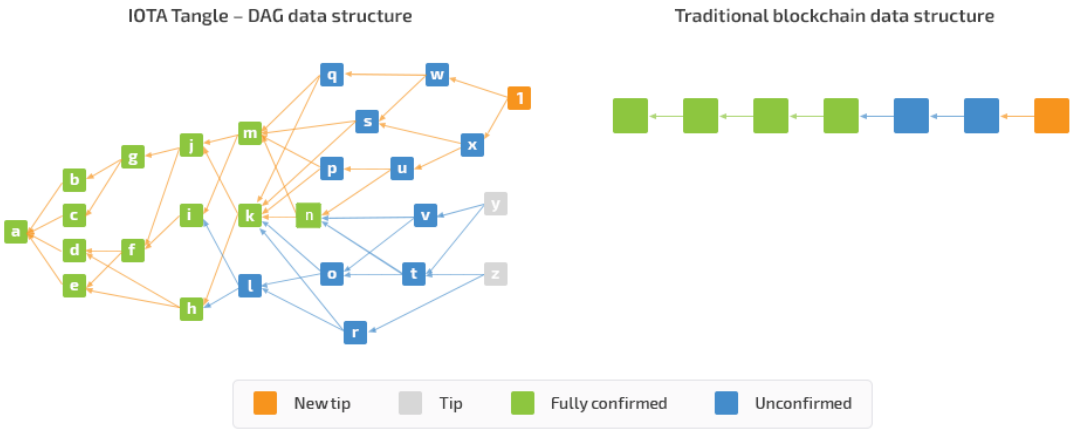

Figure 5.1: Tangle vs common blockchains representation[1].

We can observe how it differs from a common blockchain in `Figure 2.9`. Each node of the graph is a transaction:

- The transaction *'a'* is called **genesis transaction:** we said that all IOTAs have already been distributed, and the genesis transaction is the special

[1]Credits: apriorit.com

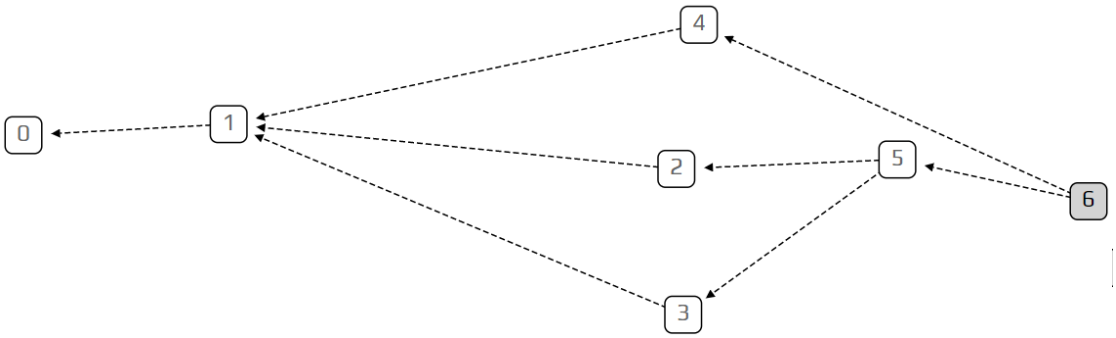

Figure 5.2: Simple representation of the Tangle[2].

transaction that dealt with the entire minting process of the IOTAs. No more IOTAs will be ever forged in the future.

- The orange and the grey transactions are called **tips**: a tip is a transaction that has not been approved by any other transactions. We say that a transaction A **directly approves** a transaction B ,if there is an oriented edge that connects A to B in this specific direction. Instead, we say that A **indirectly approves** B, if there is a path of length greater than 1 that connects A to B in this specific direction.

- We refer to green transactions as **confirmed transactions**: we say that a transaction is confirmed, if it is directly or indirectly approved by every tip of the Tangle.

- The blue transactions are **approved but not confirmed yet**: if there are 100 tips and a blue transaction is approved by 70 of them, then the transaction is considered to be 70% confirmed.

As the Tangle got inspired by a DAG model, it inherits all the properties of a graph. We define `Height` of a transaction, the length of the longest path from that transaction to the genesis transaction. If we use as example the Tangle representation in `Figure 5.2`, the `Height` of the transaction *'6'* is 4. In fact, the longest path from *'6'* to the genesis transaction *'0'*, is the one in the middle (*'6-5-2-1-0'*) that has exactly 4 links. Instead the `Height` of the transaction *'4'* is 2.

Another inherited property is the `Depth`. It is the number of transactions that are encountered in the longest inverse path that starts from the considered

[2]Source: blog.iota.org

transaction to a tip. Using the same example as before, the `Depth` of the transaction *'4'* is 2 (*'4-6'*) while the `Depth` of the transactions *'2'* (*'2-5-6'*) or *'3'* (*'3-5-6'*) is 3.

5.3 The IOTA Network

The P2P network of IOTA is composed of several nodes distributed throughout the world. A node is the core unit of the network, and it runs the software that gives reading and writing permissions on the Tangle to it. The first version of the nodes software was the **IOTA Reference Implementation**, or **IRI**, written in Java. This is now deprecated in favor of a new version, called **Hornet**, written in Go and runnable even in low-end devices such as a Raspberry Pi[20].

When a node, independently from its location in the world, issues a new transaction, it tries to forward the transaction to all of its neighbouring nodes. As in every other distributed system, this broadcasting method allows each node to validate all the transactions and eventually store them in its copy of the ledger, after having checked if they already exist in there or not; in fact nodes may have different transactions in their ledgers at any time. To ensure that this inconsistent state lasts as short as possible, each node must synchronize its state with the rest of the network. We say that a node is synchronized when it has solidified all milestones up to the latest one, which in practice it means that in the software run by a node, the two values `latestMilestoneIndex` and `latestSolidSubtangleMilestoneIndex` are the same. More details about these two values, and milestones in general, are reported in the next sections.

'Solidification' is the process in which a node asks its neighbors for the history of all milestones in the Tangle, starting from an entry point milestone and ending at the latest one. When a node has a milestone's history up to the entry point milestone, it marks that milestone as solid, and starts the process again from the next milestone. As a result, the older the entry point milestone is, the longer solidification takes.

Another interesting mechanism is the possibility of a node to create an offline version of the Tangle that would be eventually validated and attached to the online Tangle in the future. This would be useful in such those cases where it cannot be provided a stable internet connection: for example, if nodes, connected to sensors on containers being shipped, lose their internet connection when ships are in the middle of the ocean, they could create an offline version of the Tangle, and then submit it to the real one, once the connection is restored. This process is called **partitioning**.

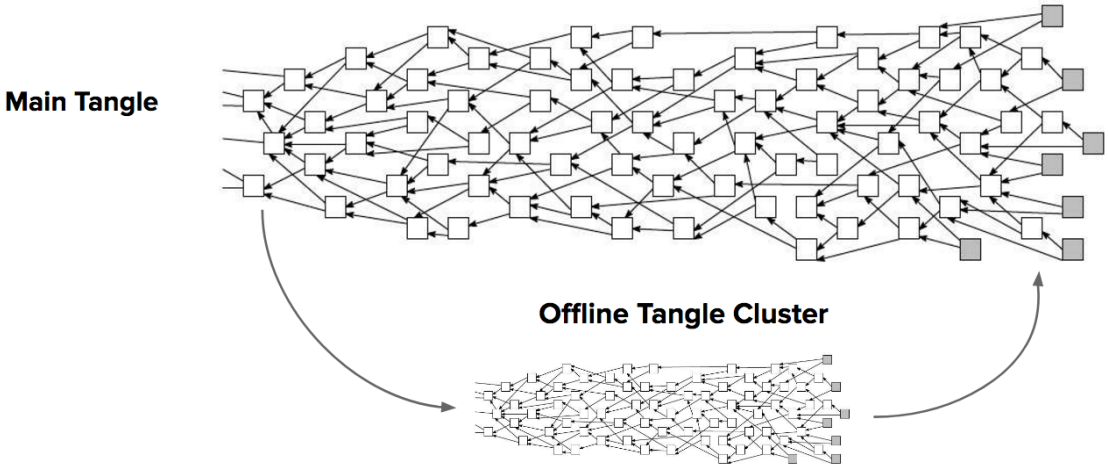

Figure 5.3: Offline transactions (Partitioning)[3].

5.4 Ternary Number System

When IOTA has been designed, ternary logic has been chosen to run through the entire project. Before we clarify why IOTA relies on ternary logic, we first need to take a closer look at the two systems that are relevant to this document.

- **The binary system.** A **bit** (a binary digit) can assume exactly two states: 0 and 1. Eight bits make a **byte** that can represent $2^8 = 256$ different combinations.
- **The ternary system.** A **trit** (a trinary digit) can assume exactly three states: $-1, 0, 1$ (also known as balanced ternary system). Three trits make a **tryte** that can represent $3^3 = 27$ different combinations.

The opposite number of each balanced ternary digit, is determined by swapping each -1 by 1 and vice versa. Thanks to this, negative numbers can be represented just as easily as positive ones: in fact no negative sign needs to be noted, such as the decimal system, or particular method to represent negative numbers, such as the two's complement as in the binary system.

This circumstance makes some calculations in the ternary system more efficient than in the binary one. Since trytes are even more complex than bytes, it is

[3]Source: blog.iota.org

important to make them more readable. This is done by converting them into some kind of other language. For this purpose, the *IOTA Development Team* has created the tryte alphabet. This consists of the number 9 and the capital letters $A - Z$. This makes a total of 27 different digits, exactly the number of combinations of a tryte. Thus, each combination of a tryte can be represented by a digit.

Tryte	**Dec**	**Char**		**Tryte**	**Dec**	**Char**
0,0,0	0	9				
1,0,0	1	A		-1,-1,-1	-13	N
-1,1,0	2	B		0,-1,-1	-12	O
0,1,0	3	C		1,-1,-1	-11	P
1,1,0	4	D		-1,0,-1	-10	Q
-1,-1,1	5	E		0,0,-1	-9	R
0,-1,1	6	F		1,0,-1	-8	S
1,-1,1	7	G		-1,1,-1	-7	T
-1,0,1	8	H		0,1,-1	-6	U
0,0,1	9	I		1,1,-1	-5	V
1,0,1	10	J		-1,-1,0	-4	W
-1,1,1	11	K		0,-1,0	-3	X
0,1,1	12	L		1,-1,0	-2	Y
1,1,1	13	M		-1,0,0	-1	Z

Table 5.1: Ternary Alphabet

5.4.1 Advantages of a ternary system

When mechanical calculating machines were replaced by electrical ones, the main components were relays, and they could only assume the states “on” and “off”. Then, relays have been replaced by transistors that still assume the states “on” (voltage “on”) and “off” (voltage “off”), but characterized by a better fault tolerance and efficiency.

Today’s computers consist of many interconnections and components that are used to transmit and store data, and to communicate with other components. These still use the binary system with the two clearly separated states *on*=1 and *off*=0. Binary systems are still used because of their high speed in changing circuit states. Transistors are very fast and effective switches. However there is a disadvantage that comes from the use of transistors themselves when it is needed to increase the

speed of a chip. To do so, more transistors have to be built on the chip by making it larger or by reducing the size of the single transistor. Currently the common trend is the second one, but it causes the disadvantage of higher heat generation and greater susceptibility to faults.

Using a system with three states would have advantages because it could rely on other basic elements instead of transistors to produce more efficient integrated circuits. Ternary is more efficient because it has the highest density of information representation among other integer bases. Thus, in the ternary system, larger numbers can be accommodated in less memory. For example, the decimal number 6 in the binary system would be the number 110 (needs 3 digits), while in the ternary system would be 20 (one digit less). The efficiency of a numbering system to the base of 3 is more efficient about 1.58 times than the one to the base of 2. In other words, the ternary system permits to save extra memory and, moreover, calculations would run faster with a lower clock number of the chip.

The effort required for a ternary system to build a complex logic circuit within the CPU can be reduced to about 36% compared to an equivalent binary system. This leads to a corresponding energy saving in addition to a space-saving smaller design of the microcontroller. However ternary boards have not been used in the computer industry yet, because the hardware implementation is much more complex and because of the lack of mass market support[21].

5.5 Seeds, Addresses and Keys

As for Bitcoin and for each other cryptocurrency, IOTA uses addresses as "bank accounts" that are associated to a particular private key from which they are generated. Whoever owns these keys, has access to the funds of each related address. Each key and so each address can be derived starting from a particular secret string called seed. An IOTA seed is a 81 character long string which maps to 81 trytes, and in fact, it is made of characters from the ternary alphabet. An IOTA Seed can be generated safely with the following command in a Linux shell prompt:

```
cat /dev/urandom | tr -dc A-Z9 | head c${1:-81}
```

To transfer IOTAs from an address to another one, it is needed to sign a transaction with the private key of the source address, proving the ownership of the funds to the other nodes. Thanks to this mechanism, addresses can be safely shared to exchange cryptocurrency, because only the seed owner knows the private key of a certain address.

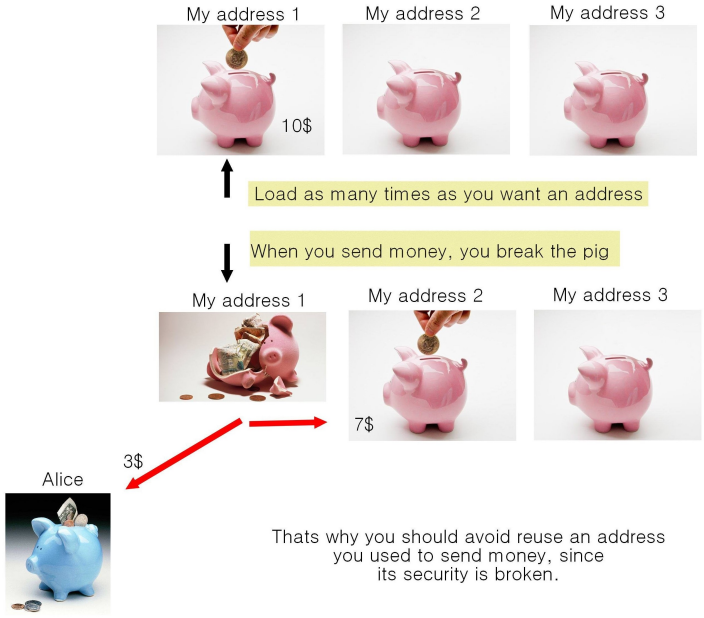

Figure 5.4: Representation of address management after a transaction[4].

An address is characterized by a specific security level and a index:

- **Index:** a number between 0 and 9.007.199.254.740.991.
- **Security Level:** a number between 1, for the lowest security level, and 3, for the greatest security level.

A tuple made of the same seed, index and security level, will always give the same output address.

Once a transaction is sent, the source address should never be reused as each output reveals part of the private key. Receiving outputs can be pooled in a single address, but once that address is emptied it shouldn't be reused for either sending or receiving. The seed is not compromised if funds are received in an address that has been already used for an output transaction, the funds are. This is due to the quantum-resistant cryptographic scheme used by IOTA, called **Winternitz One Time Signature (OTS) Scheme**.

For this reason, when a transaction with a certain source address with the index x is built, IOTA automatically generates a new address with index $x + 1$, and

[4]Source: reddit.com

moves the funds from the previous to the new address as for the parallelism of the piggy banks in `Figure 5.4`.

When an address has been used for an output transaction, the security for the next transactions is reduced and reduced. In fact, because of the use of the Winternitz OTS scheme, a part of the private key of the source address is revealed after a single output transaction. Using an address to make more than one output transaction, will compromise the security of the private key by 50% for each iteration, making the address more susceptible to cyber-attacks.

5.6 Structure of a Transaction

Every transaction in the Tangle is identified by a hash (digest). To make it simple, a transaction is like a single instruction that allows to deposit or to withdraw IOTAs from an address or just to send some data without exchange of coins. In this section we will describe the possible types of a transaction, their differences and their structure.

Until now, we referred to the nodes that compose the Tangle as single transactions (the squares represented in figure 2.9 and 2.10), but this is not really correct. Each node of the Tangle is instead a sort of "packet" that wraps a set of indivisible transactions that must be entirely accepted or rejected. These packets are called **bundles**. To explain it in an easy manner, if there is a transaction that deposits some coins into an address, this must be linked to another transaction which deals with the withdrawal of those coins from another address: these two transactions must be in the same bundle.

We now suppose that a certain bundle is published to the Tangle, and in order to make the bundle valid, it must confirm two tips as we already said. Each transaction within the bundle is made of a set of fields that defines its characteristics:

- `Hash:` this field uniquely identifies the transaction on the Tangle. This value is generated by computing the hash of the trits of the transaction.

- `signatureMessageFragment:` it contains the signature or a message, which may be fragmented over several transactions of a bundle. If a transaction records a transfer of funds, then the digital signature must be applied. Instead if the transaction just contains a message, no signature is required.

- `Address:` it contains the address id related to the transaction.

- `Value:` if it is greater than zero, then the `Address` field, refers to the recipient address, otherwise if the value is lower than zero, it refers to the source address.
- `Timestamp:` Unix epoch value that refers to the publication of the transaction on the Tangle.
- `currentIndex:` it represents the position of the transaction within the bundle. If this value is equal to zero then the related transaction is called *tail transaction*, instead if it is equal to the `lastIndex` value it is called *head transaction*.
- `lastIndex:` the last position that a transaction can assume within the bundle. The total number of transactions is equal to `lastIndex` + 1.
- `bundle:` it is the hash of the entire bundle. It is generated by computing the hash of all meta-data of each transaction in the bundle.
- `trunkTransaction:` if the transaction is the *head transaction*, then this field contains the hash of the first transaction of tip 0, otherwise it contains the hash of the next transaction within the same bundle.
- `branchTransction:` if the transaction is the *head transaction*, then this field contains the hash of the first transaction of tip 1, otherwise it contains the hash of the first transaction of tip 0.
- `tag:` this is a name defined by the user to easily find the transaction in the future. This field may be empty.

In addition to this, transactions that belong to the same bundle can be of three different types:

1. **Input Transaction.** It contains the instruction to withdraw IOTAs from an address. In order to be valid it must contain: a negative value in the field `Value`, an address containing at least the import of the withdrawal and at least the first fragment of a valid digital signature in the field `signatureMessageFragment`.
2. **Output Transaction.** It handles instead the deposit of coins into another address. Transactions of this type are easily recognizable because the value of the field `Value` is always greater than zero and the address doesn't belong to the sender.
3. **Zero-value Transaction.** This transaction doesn't handle coins but just a message. The message may be fragmented so the field `signatureMessageFragment` could contain a piece of the message or could contain it entirely.

5.7 Consensus in IOTA Network

Sending a transaction could be summarized in these 4 steps:

1. **Construction of the transaction.** The node that wants to issue a transaction (your computer or smartphone or every other device that is capable of running the node software) is in charge of creating it and signing it with its private key.

2. **Tip Selection.** Once the transaction (the bundle) is added to the Tangle, it must select two tips to approve. Each selected tip must be validated to be sure that the *sub-tangle* referenced by it contains valid transactions. Then, the new transaction becomes a new tip.

3. **Proof-of-Work.** The node checks that the selected tips are not in conflict with each other, meaning that there are not double spending events. Then, a simple PoW is necessary to prove that the new transaction has been correctly validated by a node. The PoW mechanism is similar to Bitcoin's. The main purpose of it, is to force a computational cost, even if small, to emit a transaction in order to hinder spam and sybil attacks. The mechanism makes the control of the majority of the transaction of the entire network infeasible for a malicious user. In fact, computational power is a very expensive resource to obtain, as we repeatedly said.

4. **Transmission.** Finally the transaction is broadcasted through the IOTA network to the neighbouring nodes that will send the information to their neighbours and so on.

Since points 1, 3 and 4 should be now clear, we want to focus on point 2 to understand in detail how tips are selected, and why this is an essential step for consensus in IOTA.

5.7.1 Tip Selection

The process is executed by each node willing to publish a transaction to the Tangle, by using the **Weighted Random Walk (WRW) algorithm**. The goal of the WRW is to generate fair samples starting from a complex distribution. IOTA uses it to:

- Select two tips when a transaction is created, in order to check its validity.
- Determine if a transaction has been confirmed.

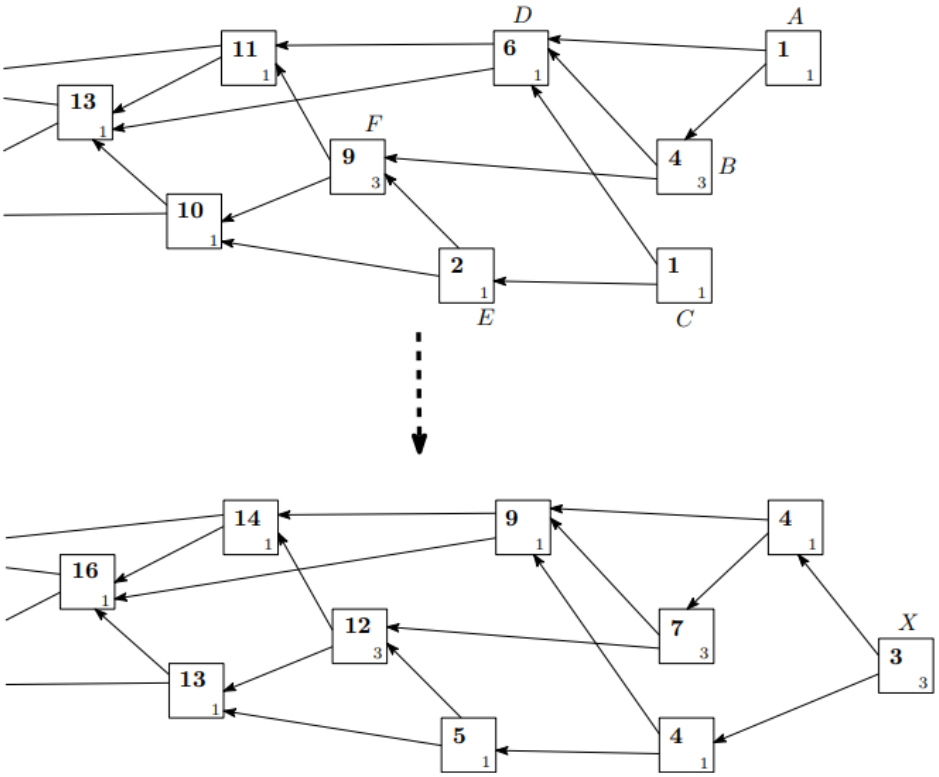

Figure 5.5: Weights and cumulative weights for a transaction[5].

The IOTA foundation chose this algorithm primarily to avoid lazy tips, since nodes could potentially approve whatever transaction. A lazy tip is one that approves old transactions rather than recent ones. It is lazy because it does not bother keeping up to date with the newest state of the tangle, and only broadcasts its own transactions based on old data. This would not help the network, since no new transactions are confirmed[22].

Each transaction (bundle) in the Tangle is characterized by a weight value and a cumulative weight value. The first is an intrinsic value of the transaction, while the second is a value equal to the sum of the weights of both direct and indirect approvers, plus the weight of the current transaction. This means that older transactions should have a higher weight than the latest ones. An example of how weights are computed is shown in `Figure 5.5`, where for each transaction, the cumulative weight (number in bold in the top-left corner), and the weight (the

[5]Source: The Tangle - Whitepaper

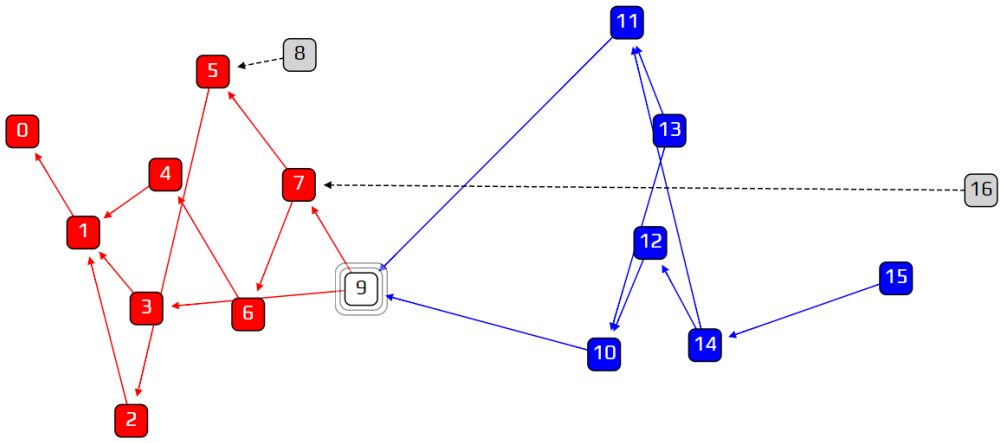

Figure 5.6: Wanted vs Lazy behaviour[6].

small number in the down-right corner) are reported.

For simplicity, IOTA attributes the value 1 to the weight of all transactions. So the formula to compute the cumulative weight of a transaction X becomes:

$$W_{cum,X} = 1 + |direct| + |indirect|$$

where $|direct|$ and $|indirect|$ are respectively the numbers of the direct and indirect approvers.

The strategy to decentivize the approval of lazy tips is to create a system that pushes nodes to use the WRW algorithm that prefers to walk towards heavy transactions (transactions with a higher cumulative weight) rather than light ones. If the majority of the nodes in the network uses this algorithm, then the others will be penalized because lazy tips are unlikely to be approved, since the walks towards them have probably a smaller cumulative weight. To better explain this behaviour, we consider the example shown in `Figure 5.6`. Transaction 16 is, in this case, a lazy tip, and in order for it to get approved, the "random walker" must reach transaction 7 and then choose transaction 16 rather than transaction 9. But this is unlikely to happen, because transaction 16 has a cumulative weight of 1 while transaction 9 has a cumulative weight of 7. This mechanism effectively discourages lazy behaviour[22].

At this point, it is important to understand that randomness is still crucial, otherwise unwanted behaviours might happen. In particular, the main problem is

[6]Source: blog.iota.org

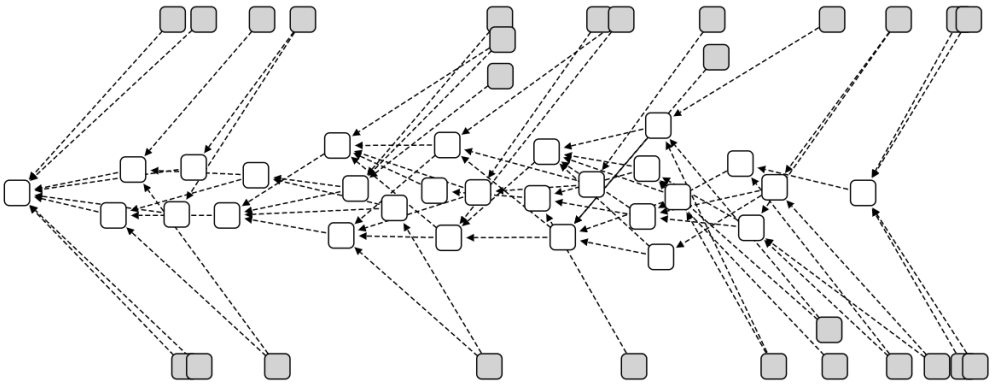

Figure 5.7: Tangle representation without using the random component for tip selection[7].

the one reported in `Figure 5.7`: if we insist on choosing only the heaviest transaction at any given point, a large percentage of the tips will never get approved, creating a corridor of approved transactions (white squares), and forgotten tips on the sidelines (gray squares)[22].

For this purpose it is essential to find a balance between the randomness and the bias towards heaviest transactions. This balance is achieved through the parameter α that sets how important a transaction's cumulative weight is. If α is equal to zero, then weights become useless, otherwise if we set α to be very high, there is no randomness. This implies that the real challenge is to fine tune this parameter. The right value of α is defined by the **Monte Carlo Markov Chain**.

Once the two tips are selected, they will be verified and eventually approved. Approving a transaction involves verifying that it does not breach the rules of consensus: in particular, that none of the accounts have negative balances and that the chosen transactions don't contradict each other.

5.8 Snapshot

IOTA is a public and decentralized platform, working for 24 hours per day and for 7 days per week, and where everyone can store an unlimited amount of data on the Tangle just by computing a simple PoW. As the stored data size increases, the need for the storage increases too, so the costs to maintain the entire system

[7]Source: blog.iota.org

will be very high. To contain the size of the entire ledge,r and consequently the expenses, nodes could execute **local snapshots**.

A local snapshot is a process that involves a single node recording the entire ledger's state in a local file. During this process, each transaction with the same address is collapsed in a single transaction with a positive value representing the summary of the entire history of that address. Zero-value transactions are discarded. As result an "image" of the positive balances for each address is stored in the file. Thanks to this mechanism, the synchronization process among neighbours is much faster because the Tangle contains way less transactions than before.

In many use cases, the IOTA Tangle data needs to be stored for a long period of time. For example, financial data must be kept for 10 years in some cases (all detailed transactions), or, for instance, all personally identifiable information (PII) must be preserved for life[23]. For this purpose, there exists a node that, instead of dealing with local snapshots, it records the complete Tangle history. In the IOTA ecosystem, we refer to it as permanode or **Chronicle** .

5.9 Coordinator and Milestones

The cumulative weight of a transaction is considered the most important metric to define the trust level of a transaction. Once this level reaches a threshold greater than 95%, the transaction can be almost certainly considered verified and approved. This means, for a hypothetical seller, that it's very hard that a certain payment that he receives is an intent to commit fraud, when this trust level is high enough.

However this is not totally unlikely to happen. In fact, if the buyer owns enough computational power to do a double spending attack, he has just to issue a new transaction where he moves his IOTAs to another address that he owns, and makes it approve two old transactions that do not approve the one in favour of the seller. At this point, using his computational power, he should just issue as many transactions as possible in order to increase the cumulative weight of the fraudulent transaction. In this way he is capable of making it trusted for the rest of the network, that will not approve anymore, directly or indirectly, the original transaction.

As for Bitcoin this is unlikely to happen, because the dishonest actor should own a huge amount of computational power, but while Bitcoin is an already proven network with a massive installed base in terms of number of nodes, IOTA, in its current state, cannot be supported by such a thing. This means that, while IOTA

is yet at an immature stage, it is more exposed to double spending attacks due to its low transaction rate of the network.

For this reason, to improve the overall security of the network, IOTA has introduced an additional consensus mechanism that is temporary, until the network will grow enough. This mechanism relies on a particular component called `Coordinator`. This component is an application that is run by the IOTA Foundation, and this is the only reason why IOTA cannot be considered as a fully decentralized system yet.

At regular intervals, the `Coordinator` sends bundles that reference and approve two new random transactions in the ledger. The tail transaction of the bundle signed by it is called a **milestone**. When a milestone is issued, every transaction that is directly or indirectly referenced by it, is considered 100% trusted and verified. This means that the `Coordinator` is in charge of establishing the direction of the growth flow of the Tangle.

Each transaction that precedes a milestone can be of two types: a transaction can carry a certain amount of IOTAs, modifying the balances of two or more addresses, or it can be a *zero-value transaction* that just carries some messages. A transaction of the first type, due to its nature, can cause potential troubles to the network, such as double spending attacks, and so it must be always confirmed by each node. Instead a transaction of the second type is always considered confirmed if referenced by a milestone.

Among the group of milestones, there are a few that are considered more important than the others:

- `latestMilestone`: it is the last milestone issued by the Coordinator, and its value is represented by the digest of the latest transaction that the node has received by the Coordinator. The index that refers to this milestone is the `latestMilestoneIndex`.

- `latestSolidSubtangleMilestone`: it is the digest of the last milestone, approved by the Coordinator, that is considered solid. A milestone is considered solid when it references a subset of milestones that is bigger enough to be considered secure. This value is included in each new transaction. If a transaction reports a solid milestone that it is not up to date, the transaction might be never approved. The index that refers to the solid milestone is the `latestSolidSubtangleMilestoneIndex`.

5.10 IOTA 1.5 - Chrysalis

IOTA has been criticized due to its unusual design, of which it is unclear whether it will work in practice. As result, IOTA was rewritten from scratch leading to what is called **Chrysalis update (or IOTA 1.5)**, which was launched on 21^{st} April 2021 and terminated on 28^{th} April 2021.

In this update, controversial decisions as a ternary encoding and quantum proof cryptography were left behind and replaced with established standards.

This update is the intermediate stage for the IOTA network, that lays the foundations for the removal of the `Coordinator` in the future **IOTA 2.0 network**.

With this update, the entire network becomes more efficient overall allowing for a validation of transactions boost. Here's the major changes in more details:

- **Removal of the *Winternitz OTS Scheme*:** before chrysalis update, IOTA had used this scheme because of its extreme security. However this high level of security has a cost: as we have already discussed, once a transaction is issued from a certain address `A` to another address `B`, and it is signed with the *Winternitz OTS* by the owner of address `A`, a small part of the private key is revealed. This led to using an address only once when it is used as the source of the transaction. As result, the use of the IOTA ecosystem was more complex both for the users and developers. The scheme has now been replaced by the ***Edwards-curve Digital Signature Algorithm* (EdDSA)**. This is a well trusted signature scheme similar to Bitcoin one. Thanks to this replacement, now it is possible to static reuse addresses as source of transactions.
- **Switching from ternary bundle to atomic binary transactions:** originally it was thought that ternary computations were more efficient than binary ones. It might be still true, but only under the hypothesis that computations are done by ternary based processors that don't exist in the market. So after years of use of a ternary based protocol, the *IOTA Foundation* decided to re-examine the previous choice and switch to a traditional **binary based protocol**. They also removed the concept of a **bundle of transactions**. Now bundles don't exist anymore and each node of the Tangle is a single transaction. These changes enable:
  - the reduction of validation requirements.
  - lower network overhead and stronger spam protection.
  - lower maintainability and implementation efforts of IOTA's core node software.

- **Implementation of UTXO model:** UTXO (Unspent Transaction Output) simply means that instead of keeping track of only the balances, you also keep track of the origin and the destination of them and when they are spent. Previously IOTA used a balanced model that maps every address to the associated balance: this method limits how efficiently IOTA can deal with conflicts like double spend attacks. These situations are dealt with, by using *"the heaviest subtangle wins"* similar to how Bitcoin uses the *"longest chain wins"*. Now the new UTXO model, that carries a little more detail, will help to solve these problems and improve the efficiency of the overall network with just a little cost that is the increase of the single transaction size.

The Chrysalis update also open the doors to other new possibilities:

- **Support for colored coins**
- **IOTA smart contracts**
- **Digital Assets**
- **New Developer Libraries**
- **Programmable Wallets**

5.11 IOTA 2.0 - Coordicide

IOTA is currently not fully decentralized because of of the `Coordinator`. As discussed before, this component is a security mechanism in IOTA, that helps to reach consensus in the network and keep it secure. Because of this mechanism, the Coordinator acts as a **single point of failure**: if the `Coordinator` does not work properly, neither does the Tangle.

Right now, it's not possible to simply remove the Coordinator because the absence of this component could expose the whole network to the risk of some cyber-attacks from dishonest actors. In response to this, the *IOTA Foundation* has developed a plan, called **Coordicide**, that would permit the safe removal of the Coordinator. This would be able to solve not just this decentralisation problem, but also the so-called **Scalability Trilemma**. The trilemma claims that a DLT cannot achieve at the same time **decentralisation, scalability and security** properties, but instead if two of them get stronger, the third becomes weaker.

Coordicide is a **six part modular solution**, allowing for an easy replacement or upgrade of one of these modules once it will become obsolete.

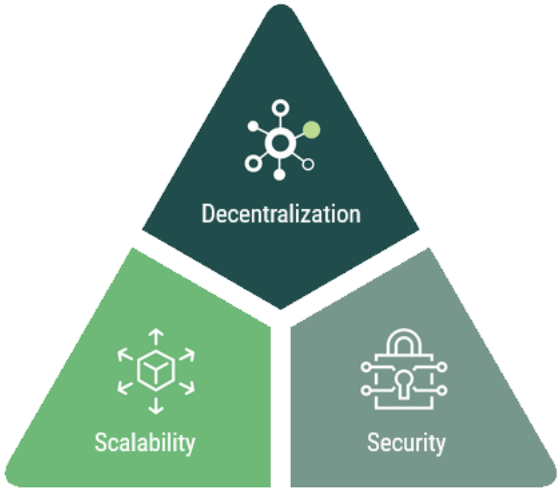

Figure 5.8: Scalability Trilemma representation. It is the most difficult challenge to face because usually, in order to improve one of these properties, a cost must be paid and this cost is often a compromise in terms of performance for one of the other properties[8].

5.11.1 Shimmer

The most important module is `Shimmer` which is a new voting scheme in the IOTA consensus layer. The name refers to a behaviour observed in nature, where swarming insects synchronize their movements to defend themselves. Without any centralized authority they decide when to change their state entirely by observing the behaviour of the near peers. `Shimmer` works in the same way just with nodes instead.

Each node query other ones about their current opinion of the ledger and adjust their own one based on the proportion of other responses they have observed. Nodes will use a voting mechanism known as ***"Fast Probabilistic Consensus"*** which is a quick and very accurate technique to see if the nodes all agree on the same state of the ledger.

[8]Credits: SEBA Bank Ag

5.11.2 Node Identities and concept of MANA

Other two modules of the Coordicide plan are about ***"Node Identities and concept of MANA"***. In a network without the `Coordinator`, a voting system such as `Shimmer` is not valid if the nodes cannot be identified. Therefore, each node generates a unique identifier that will be used to sign transactions or cast votes to ensure authenticity. However, relying on node identities alone, would make the Tangle vulnerable to sybil attacks where dishonest actors would try to control the network by forging multiple fake identities. In order to deal with this situation, IOTA introduces a reputation system called `MANA`.

`MANA` relies on the notion that reputation is difficult to gain but easy to lose. Nodes gain `MANA` by propagating valid transactions and assisting the network, but will lose `MANA` very quickly if it purposefully disagrees with the network. The more `MANA` a node has, the more trusted it is, and the more trusted a node is, the more transactions it is allowed to facilitate.

Chapter 6

BioEnPro4To

6.1 Project Purpose and Scenario

BioEnPro4To is the name of a project funded by the *Piedmont* region and *POR FESR*[1] *2014-2020* , whose architectural design and implementation are entrusted to **Concept Reply** (*Santer Reply Spa*). As reported in the official website[2], the purpose of the project is the enhancement of **bioenergy** and **bioproducts** obtained starting from integrated conversion processes of organic fraction of municipal solid waste (*FORSU*[3]), primary biomasses, residuals waste and/or other waste materials produced by the everyday life of the considered local communities.

In the following sections, a concrete solution will be proposed for one of the supply chains involved in the project: the supply chain for the production of compost. The PoC tries to solve the issues introduced in `Chapter 3`. In fact, it is important to reiterate that one of the biggest challenges in the circular economy, is the lack of an infrastructure capable of guaranteeing a safe products exchange among parties. This means that, the physical exchange must be supported at the same time by a trusted and decentralized information system that records every interaction among each-other, and that enables the well known (and already discussed) traceability and transparency features, through the immutability of data provided by a distributed ledger technology. Traceability and transparency features are the main purpose of the proposed implementation: the PoC consists of a system capable of collecting, recording and consulting tamper-proof data over the Tangle in order to provide, indeed, traceability and transparency for the supply chain

[1]Piano Operativo Regionale del Fondo Europeo di Sviluppo Regionale

[2]bioenpro4to.it

[3]Frazione Organica dei Rifiuti Solidi Urbani

processes of waste collection, treatment and recycling. The role of each component, included in the architectural design, will be discussed, focusing first and foremost to the ones belonging to the IOTA ecosystem.

6.2 Use Case Analysis

For the PoC, we have considered an hypothetical case in which three categories of actors are involved:

1. **Trucks.** Each truck of the fleet will be responsible for the day by day waste collection. Thanks to sensors installed in the vehicles, they are able to collect the required data such as the position coordinates of waste collecting points, and weight of the collected organic waste.

2. **Weighing Scales.** They include sensors to record all the amount of incoming wastes from the trucks for a specific plant.

3. **Bio Cells.** They are responsible for the waste treatments such as the anaerobic digestion process. They include sensors to monitor the process conditions such as the cell temperature or humidity.

The developed solution relies on this simplified case in which only a single stakeholder manages the three categories of actors, but the implementation has been designed in order to permit an extension of new categories and new stakeholders to interact with.

In this scenario, the system would allow each actor to independently publish their own recorded data to the Tangle, in order to store them in a safe and immutable manner. These operations are essential to avoid single point of failure issues within the system, thanks to the IOTA decentralization, and to ensure an history of the assets that can be monitored in an efficient way. In fact, technical operators belonging to whatever party of the supply chain, are allowed to easily identify issues in each stage of each process without caring about the authenticity of the data that is ensured by the underlying ledger. Moreover, logistic strategies would be enabled. Also citizens would be finally allowed to easily verify how wastes are actually treated and if the suggested ethical practices have been followed.

Finally, another benefit that indirectly offers a distributed ledger system is the fact that none of the involved stakeholders has to maintain the shared infrastructure. This guarantees a great flexibility because the system can be designed without knowing who are and who will be the stakeholders that will join the supply chain.

6.3 Overview of the System Architecture

Now that premises have been made, a deep dive into the system architecture will follow. I developed the PoC during my internship in Concept Reply. In this period, the IOTA platform has been explored and tested in the context of a supply chain for the production of compost, as well as other technologies that have allowed me to build the entire system.

The `Figure 6.1` shows a high level point of view of the BioEnPro4To architecture. Starting from the bottom, we can identify three layers:

1. **Edge Layer.** Each sensor of the actor is managed through the EdgeX framework. It is an open-source Linux platform focused on the IoT Edge Computing. With this platform it is possible to collect some data directly from the sensors and then, they are sent to the IOTA network layer, by using the **MQTT**[4] protocol, and also to the *YUCCA* platform (in the application layer). The latter is a mandatory requirement, requested by the *Piedmont* region. This layer is not part of the proposed PoC, so data of each sensor have been randomly created, to test the system.

2. **IOTA Network Layer.** The network layer, on the middle right side of the figure, is the one responsible to wrap the data, coming from the edge layer, into the structure compatible with the `IOTA Streams` framework. This is one of the two frameworks of the IOTA ecosystem used in the PoC. `Streams` is the component that allows the creation of secure channels, that are data structures that lives on the Tangle, and whose messages are somehow linked to each-other in a sequential manner. The channels are grouped in a tree structure to make them easy to navigate through. This tree structure is managed by a centralized server, that runs, under the hood, `IOTA Streams` and `IOTA Identity`. This last component is the second used framework of the IOTA ecosystem. Thanks to `Identity`, it is possible to create digital and decentralized identities in order to enable authentication and authorization mechanisms for each actor towards whatever stakeholder.

3. **Application Layer.** The last layer includes four planned applications/services. The first one is a mobile application addressed to the technical operators, allowing them to monitor and control the waste treatment process. It can work in two different ways: in the first mode, the application subscribes directly to the Tangle data streaming, while in the second one the application

[4] *Message Queue Telemetry Transport* - It is a lightweight, publish-subscribe network protocol that transports messages between devices. The protocol usually runs over TCP/IP.

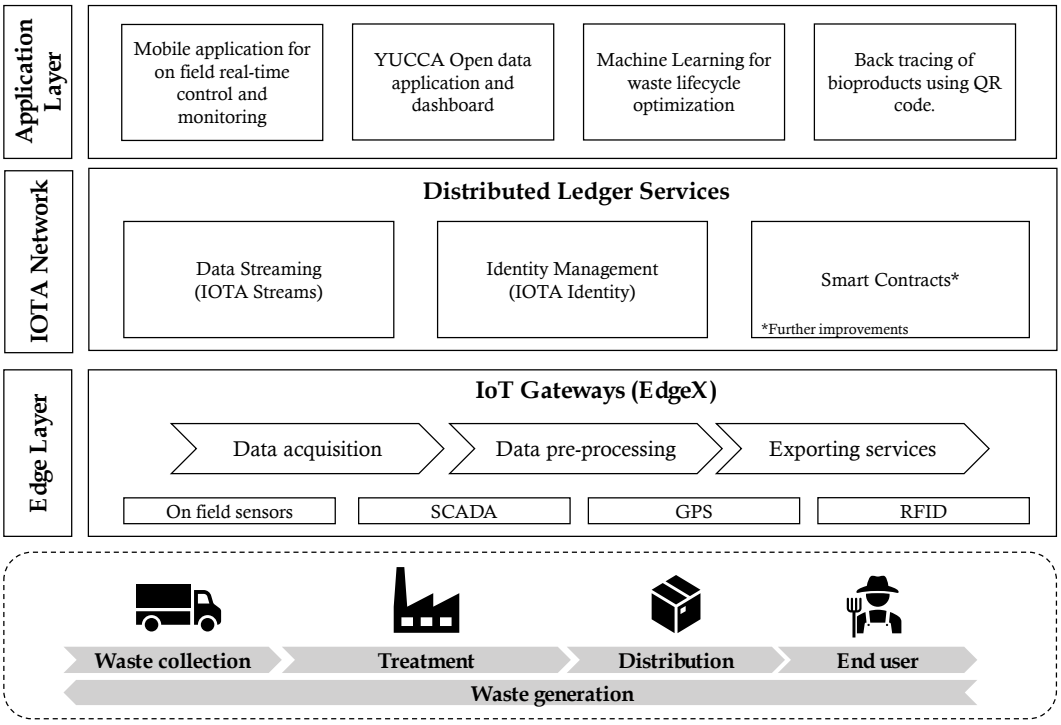

Figure 6.1: BioEnPro4To Architecture.

retrieves data from the previously mentioned centralized server that caches an updated version of the channels published to the Tangle. The second mode has been introduced to boost the performance of the application, because the mechanism to retrieve data directly from the Tangle is sequential. This slows down the user experience if there are a lot of active channels. That's why the second mode is the recommended one.

The other service is provided through the regional open data platform, *YUCCA*, where some of the processed data is published to be publicly available for third party applications. The third and the fourth applications focus respectively on machine learning algorithms, to optimize the processes involved in the ecosystem, and on services for the end-users that allow them to know in detail where the bioproduct comes from using, for example, QR codes.

However, only the first application has been developed during the implementation of this PoC. The others have been assigned to other partners of the project.

6.4 IOTA Streams

`IOTA Streams` is an organizational tool to structure and navigate secure data through the Tangle. It organizes data by ordering it in a uniform and interoperable structure and allows devices to communicate in a secure and private manner. It defines the packetization structure, message typing, and cryptographic processes. `IOTA Streams` is the natural evolution of the previous MAM (Masked Authenticated Messaging) protocol. Thanks to its flexibility it allows to build cryptographic messaging protocols, ideally with whatever transport system: currently the only supported one, is of course, the Tangle, but it has been designed in order to allow it to be extended in the future with other transportation layers such as TCP.

In this context, the Tangle can be described as a bag of unordered asynchronous messages: in order for it to be exploited as a transport layer, messages need to be extended with meta-data in order to enable packetization and sequencing. The first one is nothing but an abstraction layer for `Channels` protocol, while sequencing allows `Channels` protocol messages to be ordered in a suitable and convenient manner[24]. The mentioned protocol will be described in the next section.

The framework includes other features such as:

- Sponge-based automation for messaging processing, data encryption and authentication[5].
- Ed25519 signature scheme (RFC8032) and X25519 key exchange (RFC7748)
- Pseudo-random generator for secure key generation

This framework is currently in the **alpha stage** and it is already compatible with the new ***Chrysalis*** network.

6.4.1 Channels Protocol

`Channels` is a built-in `Streams` protocol for building secure messaging applications and it is based on the **Publish/Subscribe design pattern**. This pattern is widely used to permit asynchronous communications among different objects or processes, through an external component that acts as a transport medium (or **dispatcher**), the Tangle in this case. This means that it is not required, for the

[5]In cryptography, a sponge function or sponge construction is any of a class of algorithms with finite internal state that take an input bit stream of any length and produce an output bit stream of any desired length[24]

sender of a message, to know the identities of the potential receivers. He just has to send the messages to the dispatcher when needed. The receivers, on the other side, have to contact the same dispatcher telling it they want to *"subscribe"* to the stream of messages. This pattern solves a lot of issues that derive from the lack of knowledge: for instance, it is not required to know how many subscribers a particular channel has, or the identities of them.

In the context of the `Channels` protocol, a sender (or publisher) is named `Author` and it is the owner of the channel, while a receiver is named `Subscriber`. Moreover, there are two types of channels that the protocol provides:

1. **Single Branch Channels.** It is the most simple structure of a channel. In this structure, each message is linked to each other as a chain. This is recommended if a channel has just one publisher.
2. **Multi Branch Channels.** These kinds of channels, instead, enable more flexibility at the cost of more complexity. With this structure, different and complex use cases can be modeled. This is recommended if a channel has more than one publisher.

`Channels` is an implementation that uses the core IOTA protocol and the Tangle as a transportation and storage layer respectively. Its core functionality is achieved through the following features:

- It maintains `Streams` state through an internal link store mechanism.
- It provides numerous predefined message types (i.e. *Signed Packets, Keyloads*, etc).
- It offers decentralised transportation and storage through the usage of the Tangle.
- It provides message types for managing cryptographic access control to branches of data.
- It uses a pub/sub model with key sharing for access management.

While `Channels` exploits the Tangle to enable data integrity, the application layer itself provides authenticity and protection through direct association of data with the source, and key management solutions for assigning read/write privileges to any particular branch. Each entity that is allowed to publish within a particular branch of a channel, generates an *Ed25519* and *X25519* key pairing to be used for signing and encryption respectively. These keys are exchanged via `Subscription` messages, so once a participant subscribes to a channel, then, he can be authorised by the channel author to join in or read from any number of branches.

6.4.2 Messages Types

Different message types are needed to enable all the functionalities of the protocol. Each message will handle a specific task in the `Channels` logic, by following a predefined chain of cryptographic instructions. Each channel has its own `address` and each message has its own `message id` that identifies that message within the channel itself. In each message is embedded a field aimed at distinguishing each type of message:

- **Announce**: Once a channel is created, the `Author` must initialize it with a special message called `Announce`. It identifies the root of the channel and it will always be the first message of a channel. After absorbing the required information, the participant will verify the signature against the expected public key to ensure the message came from the expected author. Once that is verified, the multi-branch setting is detected and set accordingly in order to parse further messages. It also allows subscribers to verify the authenticity of the future messages sent by the `Author`. There is only a single `Announce` message for each channel.

- **Keyload**: Message generated by the `Author`. The `Keyload` contains a list of public keys of each `Subscriber` allowed in the channel, together with the randomly generated key for chaining (the nonce). The `Keyload` message is sent after the `Author` receives the `Subscription` messages of the users he wants to allow reading the future messages. It is used for channels configured in multi-branch mode, or to create a private channel with a predefined set of subscribers (less common).

- **Subscription**: Message sent by a `Subscriber` wanting to subscribe to a channel. If the `Author` accepts the subscription, the `Subscriber` is allowed, from now on, to read the content of future encrypted messages, by receiving the `Keyload` message.

- **Unsubscription**: Message sent by a `Subscriber` that indicates that he wants to no longer read the encrypted messages in the channel.

- **Sequence**: Special message type used as a reference pointer for other messages in a multi-branch implementation. It contains the essence necessary to derive the referenced message's identification marker for retrieval.

- **Signed Packet**: Message sent and signed by the `Author` using its private key. It is appended to the message chain within the channel. It contains two fields that wrap plain and masked payloads respectively. The message can only be signed and published by the channel owner.

- **Tagged Packet**: Message sent by the `Author` or `Subscribers`. It contains a message that is appended to the message chain within the channel. As for the signed packet, it can wrap public and masked payloads but they are anonymous because of the absence of the signature.

In `Figure 6.2` is shown a diagram of the interactions among the different message types sent by the `Author` and `Subscribers`. The figure reports a case of a double branched channel in which different subscribers choose what branch they subscribe to. Once the `Subscription` messages have been confirmed by the `Author`, two different `Keyload` messages are attached to the branches. Now `Subscribers` can read both `Signed` and `Tagged` packets in their respective branch.

Figure 6.2: Double branched channel example diagram[6].

[6]Source: legacy.docs.iota.org

6.4.3 Streams Solution for BioEnPro4To

Before explaining how the framework has been exploited to meet the project requirement, it is necessary to introduce a few problems that, at least for the used release, afflict `Streams` which are about features not working as explained in the official documentation[7]. The version of the used library is the **release v.1.1.0**[8] that is compatible with the *Chrysalis* network of IOTA, which is the one the *Mainnet* also relies on.

The following lines will describe a few problems, that pushed me to find and implement a workaround, together with the related solutions:

- **Limited Actions.** The mechanism proposed by the IOTA Foundation for the channel protocol is very tricky and limited when the complexity of the use case increases. For instance, for a multi-branch channel, there is, currently, no way for an `Author` to approve a set of subscribers for a branch A and a different set for a branch B. Moreover, once a `Keyload` message is sent by the `Author`, and few packets are appended to the `Keyload`, new `Subscribers` willing to subscribe to that branch, are not able to do it.

  I report an example to better understand the flow. We take as reference the `Figure 6.2`, and we suppose that an author `A` creates a double branched channel and initializes it with the `Announce` message. Now we suppose that there are two subscribers, `S1` and `S2` that want to subscribe to the branch `B1` and `B2` respectively. The steps to follow in order to achieve this, should be:

  1. `S1` and `S2` send their subscription messages linking them to the `Announce` message.
  2. `A` receives the subscriptions and stores the subscribers keys in order to allow them to read the future packets. Then he creates two `Keyload` messages to start two different branches for different packet chains.

  But here is where the problems arise. First of all, there is no way to understand for `A` to distinguish what subscription is linked to `B1` or `B2`, because both are appended to the Announce message (and this is the only way to do so). Then, once `A` has stored the subscribers information into its state, `Keyload` messages always wraps all the information with no mechanisms to actually select which subscriber is allowed to read one or another particular branch. This means

[7] legacy.docs.iota.org/docs/iota-streams/1.1/overview

[8] github.com/iotaledger/streams/releases/tag/1.1.0

that, the scenario reported in the figure, is simply unfeasible with the current implementation, because if the `Keyload` messages are sent after the reception of the subscriptions, then `S1` and `S2` will have the same permissions for both `B1` and `B2`. Otherwise if the first `Keyload` message is sent after the first subscription, and the second after the second one, we would obtain a situation where the branch, whose root is the first `Keyload` message, will be accessible just by `S1`, if we suppose that he is the first who sent the subscription. The second branch, instead, will be accessible by both subscribers because once `A` owns the subscribers information, he cannot choose to incorporate just a subset of them in the next `Keyload` message.

Finally, as we mentioned a few moments ago, new subscribers are not able to join not a single branch started by a `Keyload` message, because it wraps just the information of the subscribers that sent their request before the `Keyload` itself. This would not allow, for instance, to end-users to check how the wastes are being treated, or to new technical operators, to check the process states along the supply chain.

Chosen Solution. Knowing these issues that derives from a tricky management of different message types, and knowing that each channel does not require multiple publishers, as each actor involved in the analyzed scenario would manage its own channel, a single-branch channel structure has been chosen and the only message types that are used in this PoC are the `Announce` message and the `Signed Packet`. This choice keeps the channel structure as simple as possible and at the same time more flexible and suitable for external mechanism integrations as will be later explained.

- **No difference for Public and Masked payloads.** Public and masked payload fields provided by the `Signed/Tagged Packet` seem not to work as intended. Once the subscribers send their subscriptions, and the author sends the `Keyload` message after having processed them, both public and masked payloads behave in the same way: they can be both accessed only by the authorized `Subscribers`. With this behaviour, there is no reason to use one instead of the other.

  This hinders the possibility to maintain on a single packet data with different levels of reading permissions, which is actually a huge limitation in a use case that requires managing a supply chain, where data cannot be just either public or private.

Chosen Solution. In order to obtain the expected behaviour, an external encryption mechanism for the masked payloads is provided. Each author of a channel, and so each actor of the supply chain, is now able to build packets that wrap both public and encrypted data. The public portion of the data is readable by anyone who wants to access the channel, while the encrypted portion of data can be read just by those who know the encryption key. The chosen encryption algorithm for the masked payload is Xchacha20-poly1305.

In this way the protocol doesn't need `Keyload` messages anymore. The cost for this choice will be to implement an external key distribution mechanism to actively exploit the encrypted data portion.

- **Channel restoration.** There isn't a simple way to restore an existing channel after the application that manages the author stops. The only way to do so, seems to be the binary serialization of the author data structure in a moment when the `Announce` message is already sent, otherwise some errors are thrown by the library. This implies that in order to reuse channels, these states must be stored safely somewhere and they should be retrieved in an easy manner.

  Chosen Solution. To provide a friendly and immediate solution to the problem, the binary serialization of the author state is appended to the Announce message within a packet and carefully encrypted in the masked payload field. The structure of the message, that we can call `State Message` from now on, is reported in the following scheme:

```
SignedPacket{

  public = byte[]("<channel_id>:<announce_id>.state")

  masked = enc(serialized_binary_state, password)

}
```

All these changes made to the official `Channels` protocol, have been implemented in a library, called `iota-streams-lib`[9] using the **Rust programming**

[9] github.com/lore-lml/iota-streams-lib

language[10]. The library implements the lowest level of the network layer of the PoC. To summarize it provides the following features:

- Create single branch channels.
- Publish signed packets to the Tangle.
- Each packet is split in two parts:
  1. A public part that can be read from anyone.
  2. A masked part that wraps optionally encrypted data accessible to anyone who knows the encryption key.
- Restoring channels to keep chaining messages to an already existing channel, even after the application stops.
- Receive signed packets from a channel.

6.4.4 Usage of `iota-streams-lib`

The library has been developed in order to guarantee a simplified experience, compared to the official library, for developers that want to use this kind of solution based on IOTA Streams. The library provides two main structures, `ChannelWriter` and `ChannelReader` that replace, or rather wrap, the author and the subscriber of the channels protocol respectively. They have been modeled as follow:

```
pub struct ChannelWriter {
    author: Author,
    channel_address: String,
    announce_id: String,
    last_msg_id: String
}

pub struct ChannelReader {
    subscriber: Subscriber,
    channel_address: String,
    announce_id: String,
    unread_msgs: Vec<(String, Vec<u8>, Vec<u8>)>,
}
```

[10] rust-lang.org

The usage of these two structure is quite simple:

- To create a `ChannelWriter` to start pushing packets to the Tangle a builder pattern is used:

```
let mut channel = ChannelWriter::builder()
                    .node(node_url)
                    .build();
let (channel_id,announce_id) = channel.open_and_save(password)
                    .await?;
```

  The first instruction initializes the structure. By setting the `node_url` it is possible to choose which IOTA node to connect to. Picking a node also means to choose if we want to connect to the *Mainnet* or to the *Testnet*. The second instruction publishes instead the `Announce` message together with the `State` message over the Tangle. As response, it returns the ids of the channel and the `Announce` message. These two data must be provided to the subscribers in order to let them read each message of the channel.

- There are two ways of sending signed packets over the Tangle. The first one is to send raw data in a binary format, while the second one allows serialization in a JSON format:

```
// RawPacketBuilder to serialize and deserialize in bin
// format
let bin_packet = RawPacketBuilder::new()
    .public(&p_data)?
    .masked(&m_data)?
    .key_nonce(key, nonce)
    .build();

// JsonPacketBuilder to serialize and deserialize in JSON
// format
let json_packet = JsonPacketBuilder::new()
    ...
    .build();

let msg_id1 = channel.send_signed_packet(&bin_packet).await?;
let msg_id2 = channel.send_signed_packet(&json_packet).await?;
```

Packets are built using the builder pattern again. The `key_nonce(...)` method is used to set the encryption key and nonce. Then the packets can be sent in the same way by calling the `send_signed_packet()` method that returns the id of the sent message.

- To restore instead an existing channel the following function is used:

```
let channel = ChannelWriter::import_from_tangle(
                  channel_id,
                  announce_id,
                  psw,
                  node_url
                )
                .await?;
```

- To create a `ChannelReader` to start pulling packets from the Tangle:

```
let mut channel_reader = ChannelReader::builder()
                     .node(node_url)
                     .build(channel_id, announce_id);

channel_reader.attach().await?;
```

As before, the first instruction uses a builder pattern to initialize the structure. The second one, instead, attaches the subscriber to the channel in order to receive the `Announce` message.

- Finally to receive packets from the channel:

```
let msgs: Vec<(String, Packet) =
     channel_reader.fetch_parsed_msgs(key_nonce).await?;
```

With this method, a vector of tuples containing (`msg_id`, `packet`) is returned.

6.5 IOTA Identity

There are three levels of privacy when interacting on the internet:

- **Full Privacy:** neither parties, nor observers, can identify the interacting parties.
- **Verifiable Identities:** parties can trust each other, because they can both provide proof about their identities.
- **Pseudonymity:** both parties recognize each other through a pseudonymous identifier.

Pseudonymity is often the default setting of the Internet. However, today these limited data are more than enough for big tech companies, such as Google or Facebook, to be linked to real world identities. These associations have become extremely valuable for advertising strategies, internal studies etc.

Internet users have developed a need to identify themselves online and share their experiences and personal information with each other. However, this trend only serves the big tech companies. In fact, the validity of this mechanism has been so extensively proven that many internet services rely on them as a definitive representation of their users, but profiles are not really verified. This means that some situations, such as impersonation or fraud cases, still remain threats.

Digital identities could be the way to bridge the gap between the internet and the real-world. In such a scenario, users will be able to prove their own identities to be perfectly accurate. Moreover, with digital identities, a user can decide what information to share and with whom they would like to share it. This will maintain and even improve people's online privacy, while allowing many new features and new business opportunities[25].

6.5.1 Unified Identity Protocol

Unified Identity Protocol (**UIP**) is an implementation of digital identities on IOTA, based on the standards proposed by **W3C**[11]. UIP permits the creation of new digital identities to **anyone** or **anything** at any time. To achieve the goal, a **Decentralized Identifier (DID)**[12] is generated that serves as a reference to

[11]World Wide Web Consortium

[12]DIDs are URIs that associate a **DID subject** with a **DID document** allowing trustable interactions associated with that subject. - Source: w3c.github.io/did-core

a **DID Document**[13]. It contains public keys, and other mechanisms, to enable the subject to prove their association with the DID. However a DID alone is far from being enough to reveal useful information about the subject. Indeed, it has to be combined with **Verifiable Credentials**. These are statements about the considered subject, owner of the DID. They can be shared and verified online in a **BYOI (Bring Your Own Identity)**[14] manner, and the DID owner remains in complete control of the process.

Currently the IOTA Identity framework is in the **beta stage** and it is compatible with *Chrysalis* network.

6.5.2 The Roles of Digital Identities

As reported in `Figure 6.3`, IOTA `Identity` involves the use of three different roles:

- **Holder**: It is the owner of a digital identity and it often coincides with the subject of the DID. They generate the DID and cryptographic keypairs. Their personal data and private keys are under their own control.
- **Issuer**: It is a trustworthy party on a specific topic (such as governments). They have their own digital identity. The `Issuer` provides to users with a digital identity, verifiable credentials for different needs, which they digitally sign using its private key.
- **Verifier**: A `Holder` shares their verifiable credentials with a `Verifier` to prove a statement about themselves. The `Verifier` is able to verify the credentials by performing the following verifications:
  - **Data Integrity:** Is the credential unaltered?
  - **Issuer Trust:** Do I trust this `Issuer` to provide these credentials?
  - **Signature Verification:** Are the credentials signed by the expected parties?
  - **Validation:** Is the credential still valid?

[13]A set of data describing the DID subject, including mechanisms, such as cryptographic public keys, that the DID subject or a DID delegate can use to authenticate itself and prove its association with the DID. - Source: w3c.github.io/did-core

[14]BYOI is a form of federated identity where access to different service providers' services is permitted using credentials provided by a third-party identity provider, not credentials created for the service itself. - Source: ubisecure.com

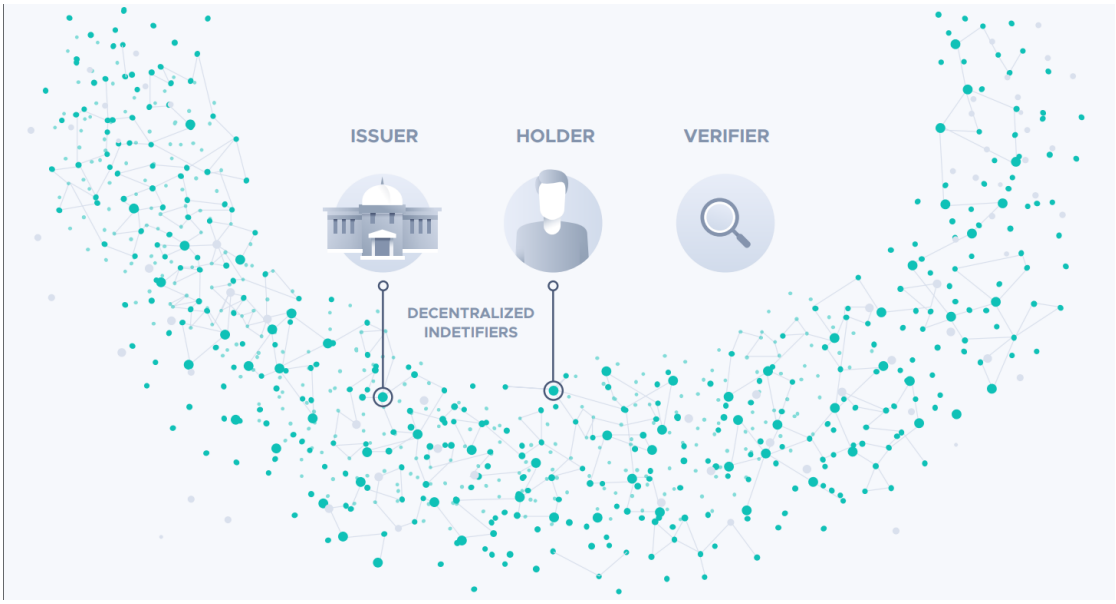

Figure 6.3: Roles of IOTA Identity. DID documents of `Issuers` and `Holders` lives in the Tangle in order to provide a decentralized, safe and public key registry[15].

6.5.3 Using Digital Identities

`Holders` can easily build up online profiles by using previously collected verifiable credentials from organizations they interact with and trust. By collecting verifiable credentials, `Holders` have more control over their personal information so that they can choose what information they want to share on the internet.

When a `Holder` asks for a new crediantal, he first has to verify himself to the `Issuer` by logging into the Issuer's environment. He shares his DID with the `Issuer` and requests the credential. The `Issuer` signs the credential together with the statements about the `Holder` with a cryptographic keypair registered in its own DID Document. At the end, when a `Verifier` needs to know specific information about the `Holder`, the latter can choose to send to it those specific attributes. The information can be quickly exchanged through various technologies such as **NFC, Bluetooth, QR Code, the Tangle, etc**. Then, the `Verifier` decides if they trust the credential's `Issuer` and verifies their signature on the Tangle.

[15]Source: files.iota.org

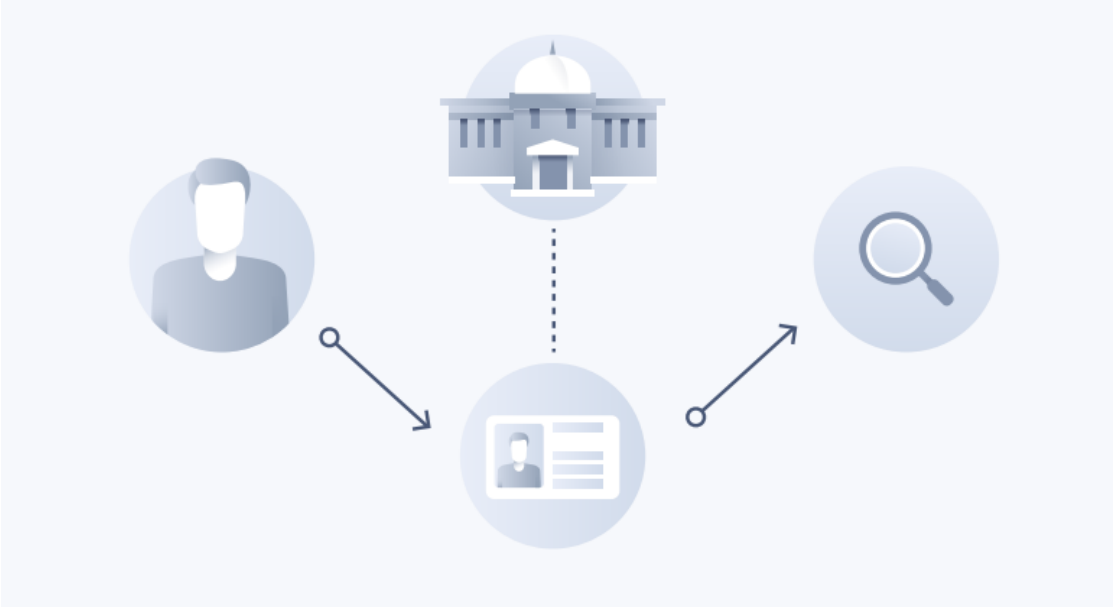

Figure 6.4: The `Issuer`, in the middle, issues a verifiable credential to a `Holder`, on the left, and then the `Verifier` check the credential by verifying the signature of the `Holder` and the DID document the credential references to[16].

Using cryptographic techniques, users can choose to share as little information as possible. For example, instead of sharing a full copy of a driver's license, the user need only prove that they own a driver's license[25].

6.5.4 Considerations

The network is designed for both humans and devices, providing a platform for trusted communication between individuals, organizations and things. Within the `IOTA Identity` framework, the Tangle is used for the following functionalities:

- **Public Key Registry:** The Tangle enables a decentralised public key registry for `Issuers` and eventually `Holders`, using *DID standards*. This allows `Verifiers` to verify credentials of a certain identity without relying on a centralized server. The *DID standards* also adds service endpoints, extending the usability of identities beyond a public key registry, to, for example, registering verifiable credential standards.
- **Revocation:** A verifiable credential can be revoked, meaning it will no longer

[16]Source: files.iota.org

be able to pass verification. The revocation is immutably stored on the Tangle, making sure no `Holder` can attempt to use their revoked credentials.

The UIP will ensure compatibility with global privacy and data management legislation. **GDPR**[17] states that citizens have the right to provide and retract consent for the storage of **PII (Personally Identifiable Information)**. Since any data stored on a DLT is immutable and cannot be removed, a DLT and digital identity solution is only **GDPR compliant** if it never stores PII on the ledger[25].

According to the GDPR Art. 4 (1), **PII is defined as**:

> *"**Any information relating to an identified or identifiable natural person** ('data subject'); an identifiable natural person is **one who can be identified, directly or indirectly**, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person"*

	GDPR-compliant possible use	**Used by UIP**
(Asymmetric) Encryption of PII	No	No
Hash of PII	No	No
Salted hash of PII	Maybe	No
DIDs	Yes	Yes

Table 6.1: IOTA interpretation of the GDPR directives for the UIP.

6.5.5 Purpose of Digital Identities in BioEnPro4To

We have taken into account the integration of `Identity` for the BioEnPro4To project, in order to enable more benefits from the decentralization that the IOTA ecosystem offers. In this case, digital identities would enable authentication and authorization mechanisms for each actor involved in the supply chains. This means that, every time an actor has to prove its identity and permission levels, it can just share its previously collected verifiable credentials to any stakeholder and they can just verify its authenticity without relying on trusted third party servers.

[17]General Data Protection Regulation

For instance, we suppose that there are two stakeholders, `S1` and `S2` that are responsible for the management of the truck and weighing scale actors respectively. Each truck and each scale would own a digital identity, in particular a DID document, and a collection of verifiable credentials digitally signed by an `Issuer` that, in this case, is one of the two stakeholders. These verifiable credentials would allow, for instance, the actors to create and manage a channel in which they can push the collected data of the process coming from their sensors.

So, once a truck `T` managed by `S1` has to interact with a weighing scale `W1` managed by `S2`, it can be authenticated in terms of actor identity and actor permissions within the system by `S2` in an autonomous manner. In fact, `S2` does not have to rely on an authentication system provided by `S1`, because they share a decentralized infrastructure, the Tangle, that provides them a public registry to achieve the goal. In this way the only requirement for certain stakeholders is to trust and to accept verifiable credentials signed by any other partner along the supply chain. Moreover, authentication and authorization mechanisms don't have to be implemented and maintained by each stakeholder, but they are designed and developed once in an interoperable way.

The same procedure, followed for `Streams`, has been used for `Identity`, and a library, called `iota-identity-lib`[18] has been developed. The `iota-identity-lib` library depends on the official `identity.rs` one developed by the IOTA Foundation, and it provides the following features:

- Creation of digital identities and store of the relative DID documents as well as private keys to manage them inside a secure vault, named **IOTA Stronghold**.
- Issuance of verifiable credentials and possibility to securely store them within a wallet.
- Possibility to verify identities and verifiable credentials.

6.6 Channel Manager

Before analyzing the detailed architecture of the entire PoC, it is necessary to describe which choices have been made regarding the management of the channels. As we discussed in the previous sections, the chosen type of channels is the single branch channel provided by `Streams`, with the addition of some external features such as an ad-hoc encryption system for masked payloads and the `State` message

[18] github.com/lore-lml/iota-identity-lib

to restore the channel. Although this modified version of channels would enable a big level of flexibility for different use cases, if it is not integrated within a more organized structure, a channel alone is a quite limited tool that is affected by three big problems:

1. The first is the fact that it is a linear chain by design. This makes random access unfeasible, and its traversal very slow. This would be a big problem for an IoT scenario, where sensors collect a great amount of data that actors keep appending to the channel, wrapping them within hundreds of packets a day.

2. The second is that, single branch channels are not meant to allow multiple publishing entities, but by using multi-branch channels we would have met more serious problems as illustrated before. This means that a single channel must be owned and managed by a single entity, which is an actor (a truck or a biocell etc).

3. The last one is about the channel address. Each channel, in order to be read, must be first identified on the tangle by the subscribers by knowing this information. So the challenge would be to design a method to easily exchange addresses of each channel in an easy and feasible manner.

To solve these issues, channels have been rearranged and modeled into a tree data structure. To avoid that a channel would include a chain of thousands of messages, or even more, it has been decided to assign to each actor one channel per day. This would make the performance problem less relevant, because the amount of data within a channel is constrained and data size does not increase after that day is passed. But using this solution the number of channels increases, so more channel addresses have to be shared.

That is where the tree model pays off: each node of the tree is a channel and each tree level contains the references of the channels of the next one. Finally each leaf of the tree represents the actual daily channel of a certain actor. Here, the real data about the supply chain processes are appended.

`Figure 6.5` shows the actual structure of the tree. The root of the tree represents the entry point for each subscriber, that now has to know only a single address to read data. The second level of the tree contains a channel for each category of actors, three in our case. The third level is made of all the actors that belong to that category. The fourth and last level, contains, instead, all the daily channels managed by an actor of a certain category. This tree represents the actual structure used in the PoC.

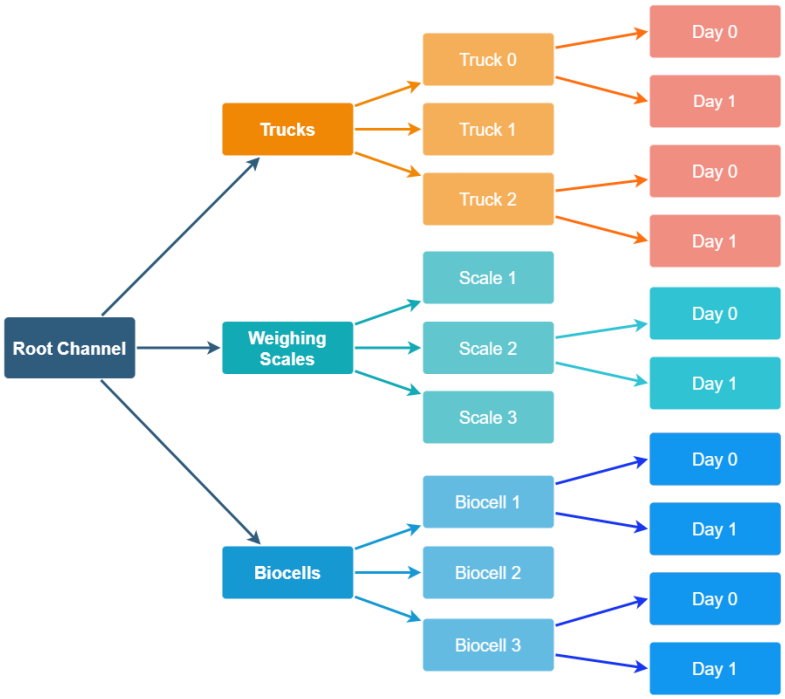

Figure 6.5: Hierarchical organization of channels.

Now that it is clear how the tree has been modeled, we can analyze `Figure 6.6` to understand what kind of information is stored within each channel of each level.

Starting from the left, we find the root channel (the entry point). In this channel are simply stored some `channel_ids` and `announce_ids`, that together compose what we have called so far `channel address`. Each channel address is linked with an identifier of a category of actors, to allow the correct tree traversal. The second column represents the second level of the tree and here, for each channel, we find a list of all the actor identifiers that belong to that category together with the channel addresses of their containers of personal daily channels.

In the next column each channel represents a container of all the daily channels of that actor. For each daily channel is reported the owner, the date, and the channel address. Finally the last column contains the actual daily channels that contain the data coming from the sensors.

`Figures 6.7, 6.8, 6.9, 6.10` show the actual data, in JSON format, that are stored on the Tangle. The images have been screenshotted from an ad-hoc web

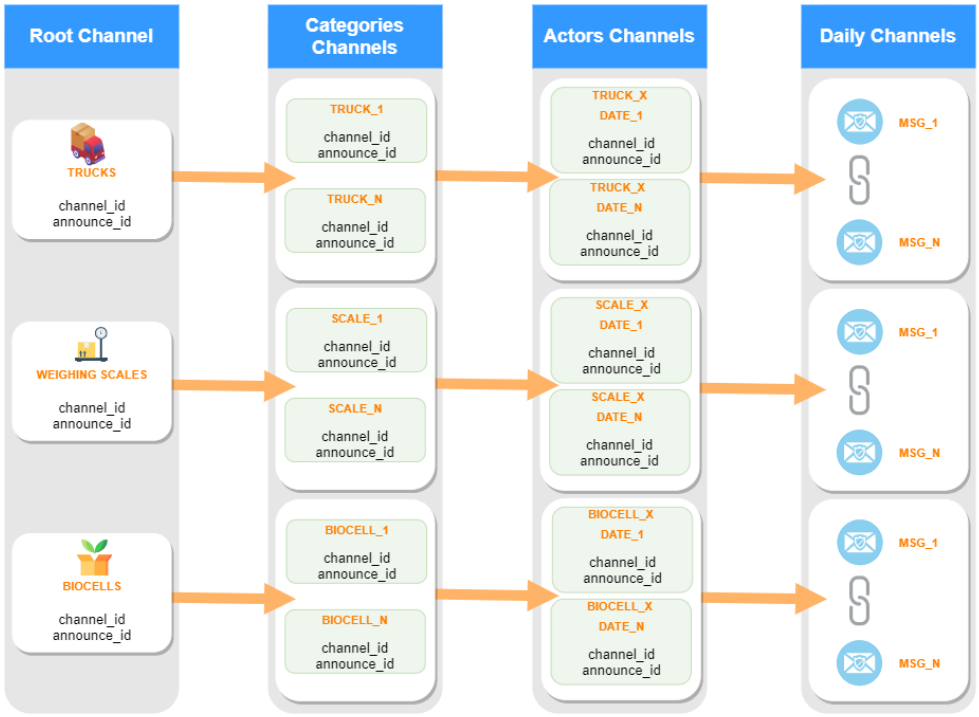

Figure 6.6: Channels details of tree structure.

application, developed for debugging purposes, that acts as a **channel explorer**[19].

The library that takes care of both initialization and management of the entire tree structure is called `bioenpro4to-channel-manager`[20] and depends on the `iota-streams-lib` library.

[19] streams-chrysalis-explorer.netlify.app/

[20] github.com/lore-lml/bioenpro4to_channel_manager

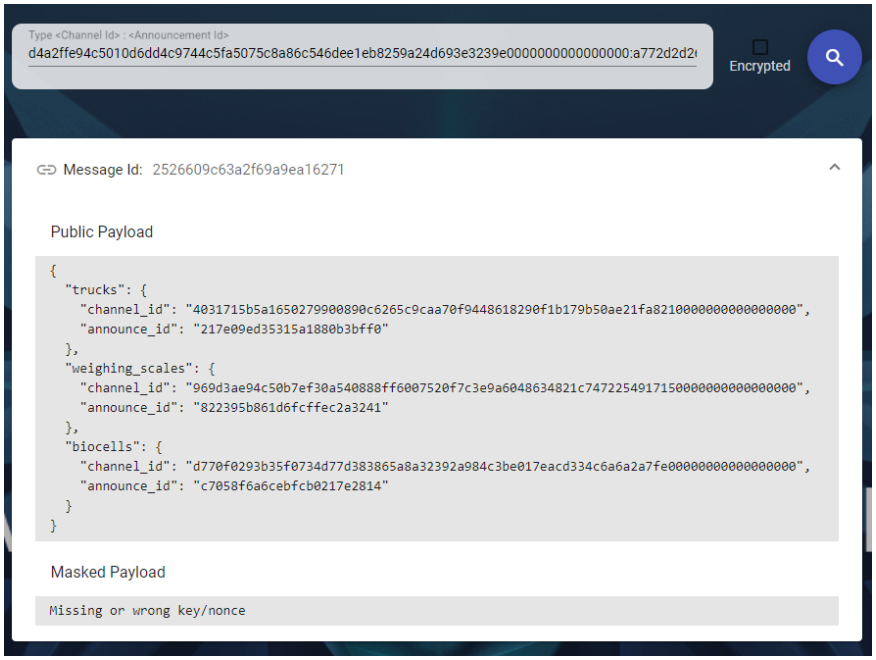

Figure 6.7: Root channel. It contains a message with the references to the category channels.

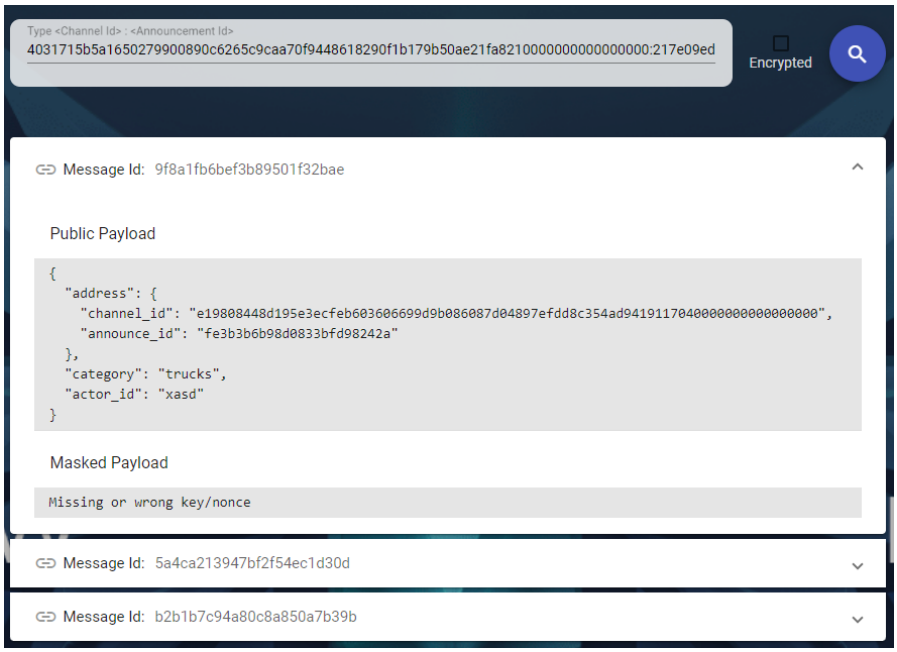

Figure 6.8: Trucks channel. It contains a message for each actor of this category. In each message different actor identifiers are reported.

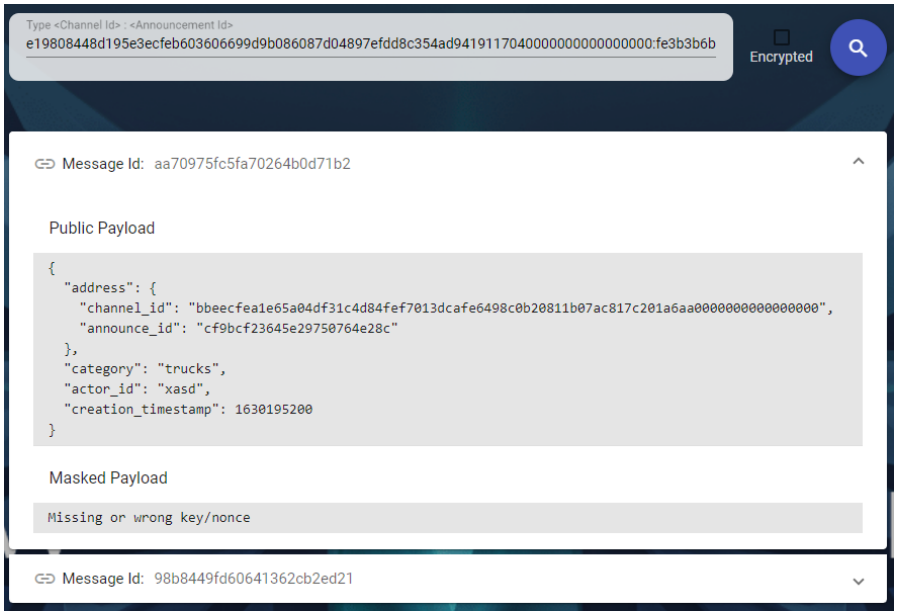

Figure 6.9: Actor channel with id `xasd`. It contains a message per each daily channel the actor manages. A timestamp is reported to indicate the date.

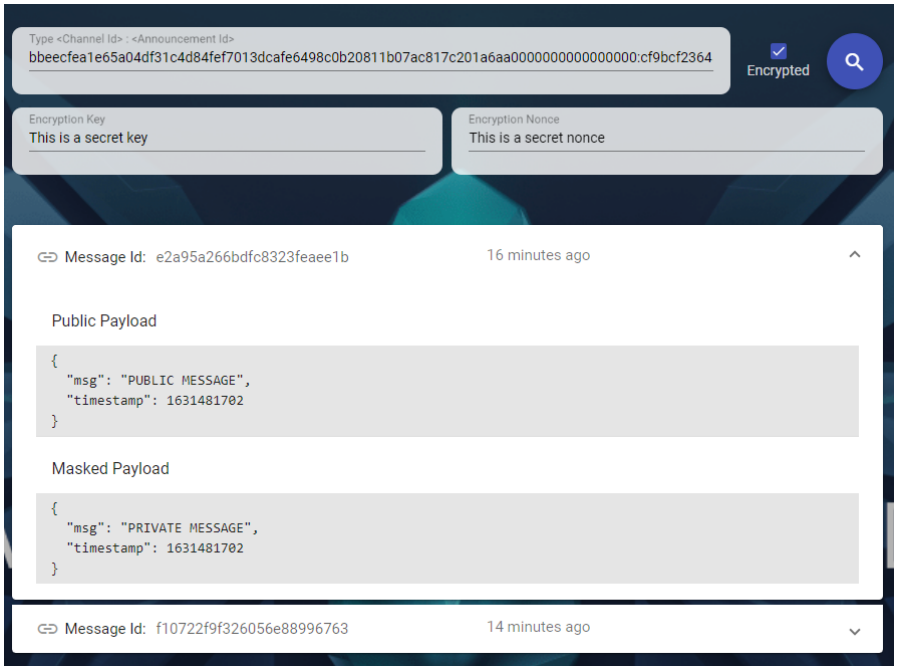

Figure 6.10: Daily channel of the actor with id `xasd` of a specific date. It contains all the data coming from the sensors.

6.7 PoC Architecture

Now it should be clear how the actors of the considered scenario, for the BioEnPro4To project, interact with the IOTA distributed ledger, so it is appropriate to start illustrating the architecture of the PoC, and how all the components interact with each other.

For a better comprehension of the architecture, `Figure 6.11` shows a representation of it, so that it can be used as the landmark for the analysis of this section.

What we know until now is that, each actor somehow collects data from its sensors through the *Edge Layer*, and then they push these data to the Tangle in their daily channels. But how do digital identities and the channel manager fit together with this?

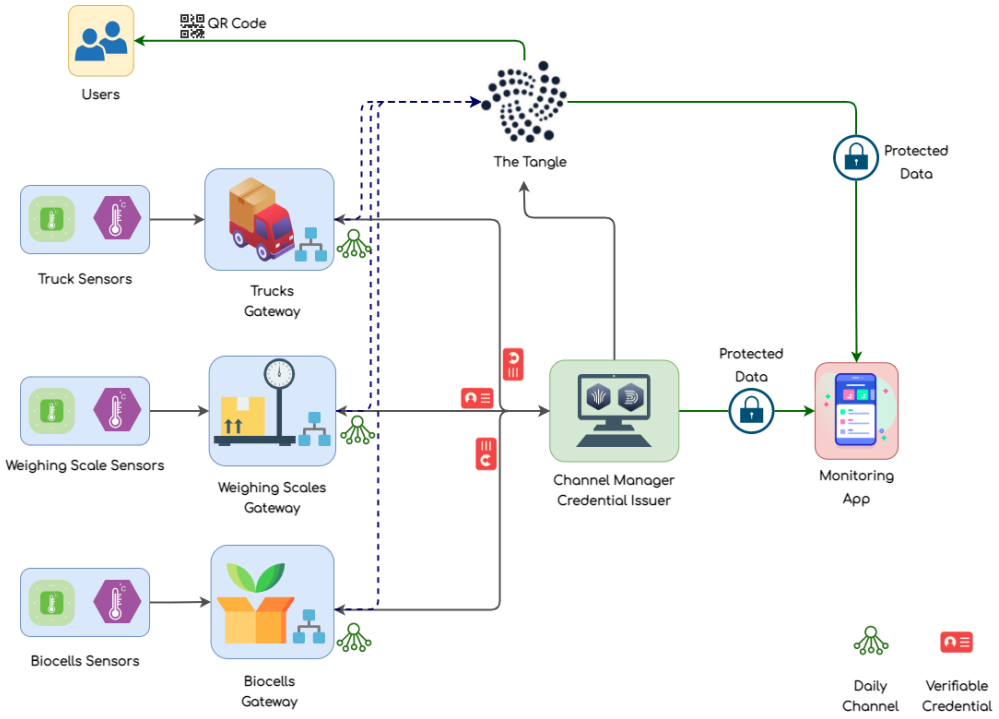

Figure 6.11: PoC Architecture[21].

[21]Icons Credits: Good Ware, Freepik, DinosoftLabs - flaticon.com

6.7.1 Server Features

The main component of the entire architecture is the one in the green box. This is a centralized server managed by the stakeholder considered for the BioEnPro4To scenario. This server has multiple roles:

1. First of all, it is the only responsible for the maintenance of the correct functioning of the channel tree structure. In other words, it has the role of **channel manager**. The server uses the `bioenpro4to-channel-manager` APIs, and it is able to initialize the tree structure in order to create traversable paths from the root to the requested daily channel, and to create daily channels on behalf of the actor who asks it to do so. When the channel manager creates a daily channel, it maintains a strict association between the channel and the authorized actor. In fact only the authorized actor will be able to access and publish data to this channel because it is the only who knows the password of the encrypted `State` message.

2. The second role is that of **Issuer**. It uses the `iota-identity-lib` library in order to issue verifiable credentials to the correct actors and check the validity of their digital identities.

3. Finally it exposes **RESTful APIs** in order to guarantee a simple interaction both for actors and for the users of the monitoring application.

The server[22] has been developed using, again, the Rust programming language and the `Actix-Web`[23] framework. The server also relies for the storage on a SQL database, `Postgres`, and every entity has been modeled as shown in `Figure 6.12`.

By watching carefully at this UML diagram, we can notice what are the entities that are allowed to interact with the server. We have the `actors` table that represents the list of each actor managed by the stakeholder. It contains the data needed to identify the actor within the system. This is a table that aims at aggregating all the common information of actors needed to identify them during an early inner stage of authentication, before obtaining a valid credential. Then, there are three more tables that model the specific data that characterize the considered actor (`trucks`, `scales` and `biocells`). On the other hand we have a `users` table. It models every user that has a particular role along the supply chain processes. For this PoC a user may be, for instance, a truck driver or a supervisor that uses the monitoring mobile application.

[22] github.com/lore-lml/bioenpro4to_gateway

[23] actix.rs

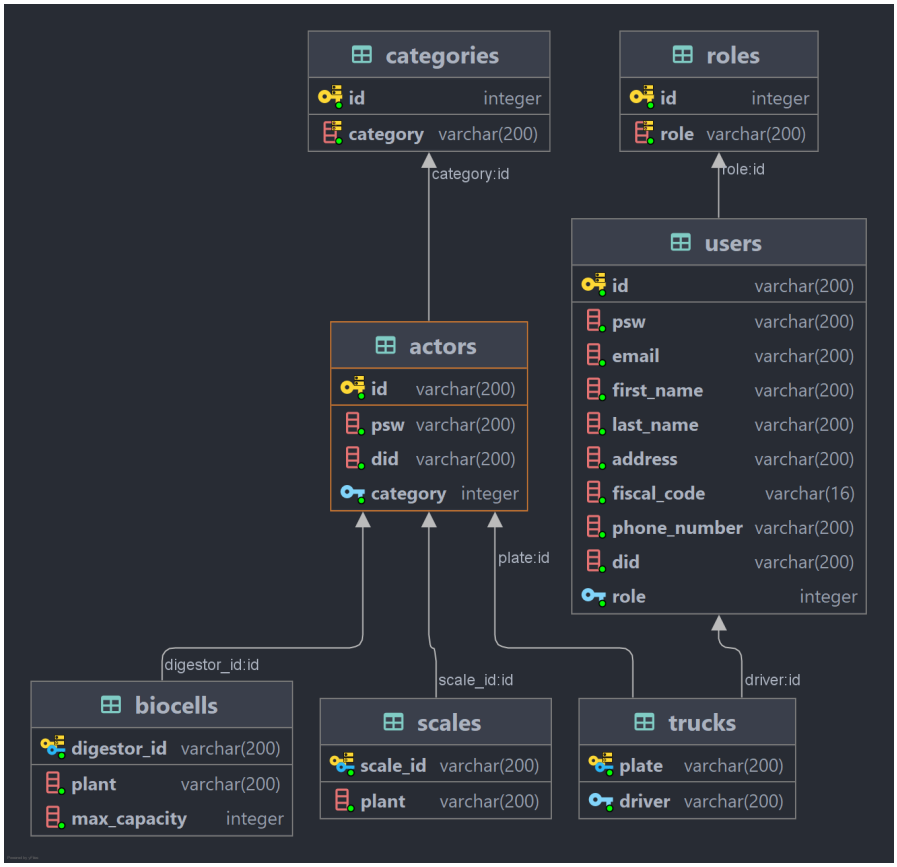

Figure 6.12: UML diagram of the entity relationships.

6.7.2 Component Interactions

At this point, we can start illustrating what is the flow of the interactions among components.

On the left of `Figure 6.11`, are reported the actors of the three different categories and their relative sensors. Each truck, scale or biocell will have a device that manages the logic of the *Edge Layer*. As already mentioned at the beginning of this chapter, the edge layer relies on the `EdgeX` framework in order to collect some data from the sensors and then, to publish those, wrapping them into several packets, to the Tangle, or rather to a daily channel. The collection process, however, is not covered by this PoC as we pointed out. Instead, what the PoC provides

to the *Edge Layer* is a **Go-lang**[24] client library, `bioenpro4to_http_client`[25]. This choice has been forced due to `EdgeX` relying on Go programming language. This library provides to these devices a simple interface to interact both with the server through HTTP calls, and with the daily channels once they have been retrieved from the server itself. To allow a direct communication with daily channels, some bindings, that allow to use from Go-lang some Rust APIs of the `bioenpro4to_channel_manager` library, has been developed by using the `Cgo`[26] package.

The first thing to do, once an actor has been *"activated"* together with its sensors and its device running the client, is to register the actor within the stakeholder system. This task is entrusted to whatever user that has that permission level. After the user is authenticated, he can register the actor to the server in order to enable, from now on, the authentication of the actor. Each actor is able to autonomously generate its authentication data such as a `DID` for its digital identity (by using, again, bindings for Go-lang of the `iota-identity-lib`), an `actor_id` and a `password`. Thanks to this, the user just needs to collect these data and forward them to the server, specifying just the category the actor belongs to. The actor is now part of the supply chain processes.

At this point, it is necessary to obtain a daily channel to start publishing data to the Tangle. To do so, the actor needs a verifiable credential that certifies that it has the authorization to create and publish data to a daily channel of a specific category. To get this credential it just has to authenticate to the server, by providing `actor_id`, `password` and its `DID`. The server checks the accuracy of the information and, if no discrepancies have been found, it issues to the actor a verifiable credential. From now on, the actor potentially has the benefit to be authenticated and authorized by any other stakeholder of the supply chain to create and manage daily channels: the only requirement is that stakeholders must trust each-other in order to guarantee the validity of these verifiable credentials that each of them is allowed to issue.

An example of an IOTA verifiable credential is reported, as a Rust struct format, in the scheme below.

[24] golang.org

[25] github.com/lore-lml/bioenpro4to_http_client

[26] Cgo enables the creation of Go packages that call C code. For this reason an additional step, to bind the Rust code to C code, has been made.

```
Credential {
    context: Url(https://www.w3.org/2018/credentials/v1),
    id: None,
    types: [
        "VerifiableCredential",
        "ChannelWriteAuth",
    ],
    credential_subject: Subject {
        id: Some(
            Url(did:iota:dev:
GC2VXM5A8CP5ozJ9R7iE8S7jrrpkSZGXmqe7NEs1Mg21),
        ),
        properties: {
            "channel_authorization": Object({
                "actor_id": String(
                    "aa000aa",
                ),
                "category": String(
                    "trucks",
                ),
            }),
        },
    },
    issuer: Url(
        Url(did:iota:dev:
EYuK6teLh6nzPAJhJx7XUX361cZNezvwj4ppq6WUFCo6),
    ),
    issuance_date: "2021-09-20T15:27:30Z",
    expiration_date: Some(
        "2021-09-27T15:27:30Z",
    ),
    credential_status: [],
    credential_schema: [],
    refresh_service: [],
    terms_of_use: [],
    evidence: [],
    non_transferable: None,
    properties: {},
    proof: Some(
        Signature {
```

```
            type_: "JcsEd25519Signature2020",
            value: Signature(
    5M9jXWidqgYZU5VbyDn6oRPfh6L5iUZp6NN5gcGxDYVSJgTdvnkwJD
    2zzPUkzB7qrTfSWgXNphNWSxxkD7YPgVpL
            ),
            method: "did:iota:dev:
   EYuK6teLh6nzPAJhJx7XUX361cZNezvwj4ppq6WUFCo6#_sign-0",
        },
    ),
}
```

Starting from the top of the scheme, it is possible to notice the `types` field, in which has been reported a custom credential type, `ChannelWriteAuth`, in order to allow stakeholders to recognize the correct type of a credential. Then, it follows `credential_subject` and `issuer` fields. In the first one, there are the DID of the subject and the properties of the credential that indicate the actual purpose of the credential. In this case it is reported that the actor with id `aa000aa` is authorized to request the creation and the ownership of new daily channels belonging to the category `trucks`. The second one, instead, just reports the did of the issuer. Finally we find the dates of issuance and expiration, which is set to a week, and the signature applied by the issuer in the `proof` field.

The actor can now ask the server to create a daily channel: to achieve this, it does an HTTP call to the server, attaching to the request the obtained credential, and providing the desired date for the daily channel together with a password to encrypt the `State` message. If the credential is valid and the daily channel does not exist yet, the server creates it and attaches its address to the channel's tree structure. In particular, once the the credential has been validated, the channel manager module of the server follows these step to update the tree structure (`Figure 6.6` can help understanding the work-flow):

1. Starting from the root, it retrieves the address of the category the actor belongs to. The category is reported in the credential, as we mentioned earlier.
2. It checks if the `actor_id` already exist in the category channel (in the second level of the tree). If not, it creates an actor channel (in the third level of the tree), and append a new packet in the category channel, referencing the new actor channel by storing its address and the `actor_id`. The `State` message of the actor channel is encrypted with a password chosen by the server, so that only it will be able to access to the channel.
3. It checks if the actor has already requested the creation of a daily channel with that specific date. If so, no more actions are needed. Otherwise, it creates a

daily channel (last level of the tree structure), and append a new packet in the correct actor channel, referencing the new daily channel by storing, again, its address, the `actor_id` and the date. The `State` message of the daily channel is encrypted with the password provided by the actor.

Once this procedure is over, only the actor is able to write into that channel, and he can request permission access to it. This permission is achieved by performing another HTTP call: the server in this case provides to the actor a binary serialization of that channel, encrypted with the same password the actor has provided during the creation. Thanks to the client library, now the actor is able to locally decrypt and deserialize the channel, so it can start publishing data to the channel autonomously.

Thanks to this mechanism, it's not required that the server manages massive requests from each actor everyday. Moreover the steps to obtain the verifiable credential and to create a daily channel have to be executed once. In fact the credentials will have an expiration time set to a week, and after two HTTP calls a day, to create and get the daily channel, each actor is independent from whatever centralized system. A detailed representation of this procedure can be found in `Figure 6.13`.

To conclude the explanation of the component interactions reported in `Figure 6.11`, there are the ones between the server and monitoring app and between the Tangle and the monitoring app (on the right). These two reflect the two modes that the app can use, reported at the beginning of the chapter.

Finally the last one, on the top of the image, would be the hypothetical feature that would allow end-users, the citizens of the Piedmont region, to autonomously check the development of the waste management processes. This has not been implemented in this PoC.

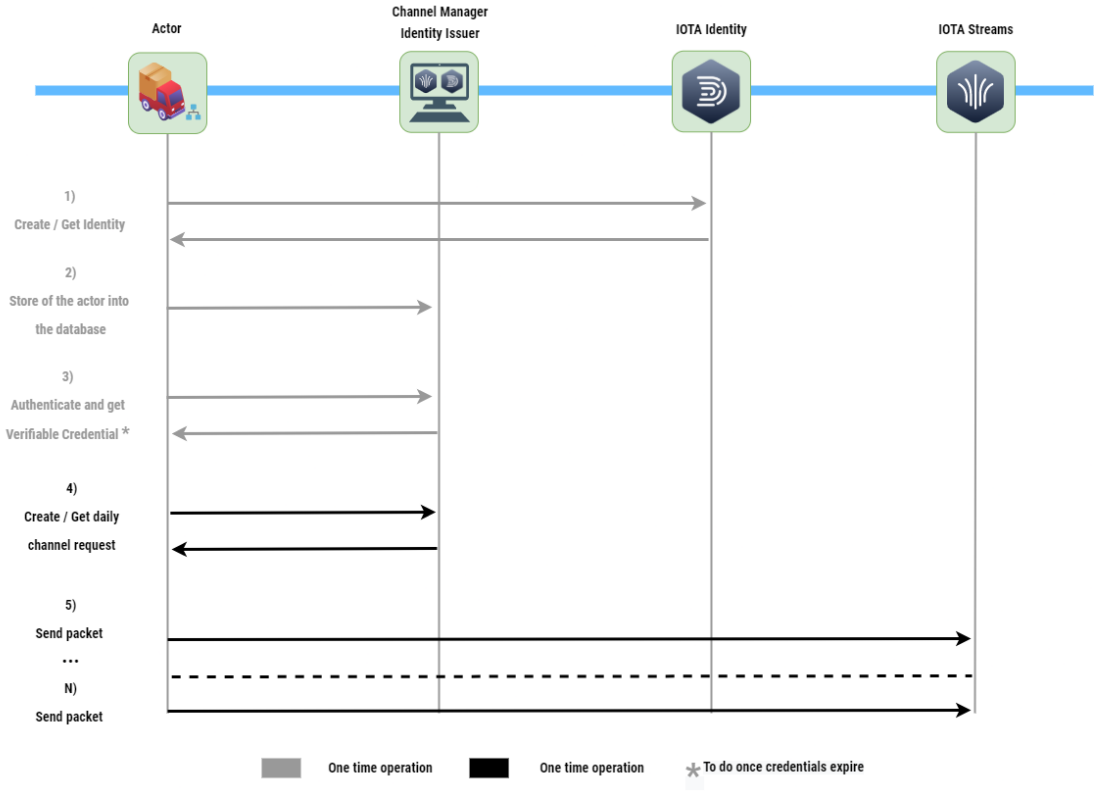

Figure 6.13: Representation of the interactions among the actor and the other components[27].

6.8 Monitoring Mobile Application

The monitoring mobile app has been designed to make the supervision experience for technical operators easier. In fact, it is important that each step of the waste collection and treatment processes would be executed without incurring in particular issues. That is because each step of the chain depends on the previous one.

The app is part of the PoC and it has been developed using the `Ionic` framework. During the research stage, `Ionic`[28] turned out to be an interesting tool to build cross-platform native apps. To achieve this, in relatively recent times, the Ionic framework has integrated a powerful and open-source runtime tool, called

[27]Icons Credits: Good Ware, Freepik, DinosoftLabs - flaticon.com

[28]ionicframework.com/

`Capacitor`, capable of building native and cross-platform apps for iOS, Android and the web, starting by a single code-base. Ionic is also really flexible, allowing developers to choose the most comfortable Javascript platform. It supports three different platforms which are `Angular`, `React.js` and `Vue.js`. For the case, Angular has been chosen for its rich set of features such as a built-in HTTP-client and routing system.

The home page, shown in `Figure 6.14`, is made of three tabs:

- The first, on the left, implements the actual monitoring features. The main page shows the daily feed of the active actors, and the possibility to check, more in detail, one of the three categories.
- In the middle, a page for some real-time stats and diagrams has been designed. The shown diagrams are interactive and allow for some filter features. The reported ones are just ideas of possible implementations.
- The last one, on the right, is just a page to check personal information and change default settings, using the special button on the top-right corner.

The most relevant features are, obviously, the one implemented by the monitoring page, that includes the actual components to interact either with the Tangle or with the server, depending on the chosen mode. When the Tangle mode is selected, the application exploits the `iota-streams-lib` library compiled to `WebAssembly`[29], to allow the reuse of the Rust implementation.

By navigating into one of the category sections, the actual view of the entire channel tree has been arranged. The `Figure 6.15` shows the content of the category channels, actor channels and daily channels. The list of the actors for that category (on the left), the list of daily channels for a specific actor (in the middle) and the list of messages of a particular daily channel (on the right).

Finally, in `Figure 6.16` are shown three more views. The first one (on the right) is the implementation of a setting section, allowing the user to select the data source (server mode or Tangle mode). The second one (in the middle) to check some warnings that the system could send on future prototypes of the system, and the third one (on the left), showing a map to track trucks during the waste collection process. These last two views are just graphical design and no actual

[29]WebAssembly (or Wasm) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications - Source: webassembly.org

implementations have been developed.

Figure 6.14: Main tabs of the home page. Starting from the left, monitoring page[30], stats page and profile page.

[30]Icons Credits: Freepik, DinosoftLabs | flaticon.com

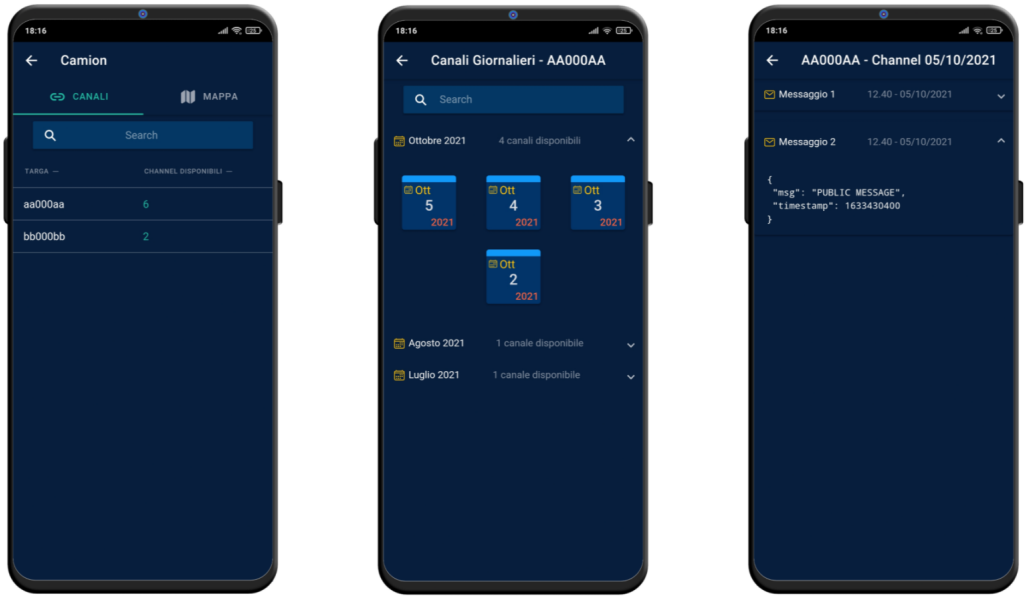

Figure 6.15: Views reporting the information of each level of the channel tree structure.

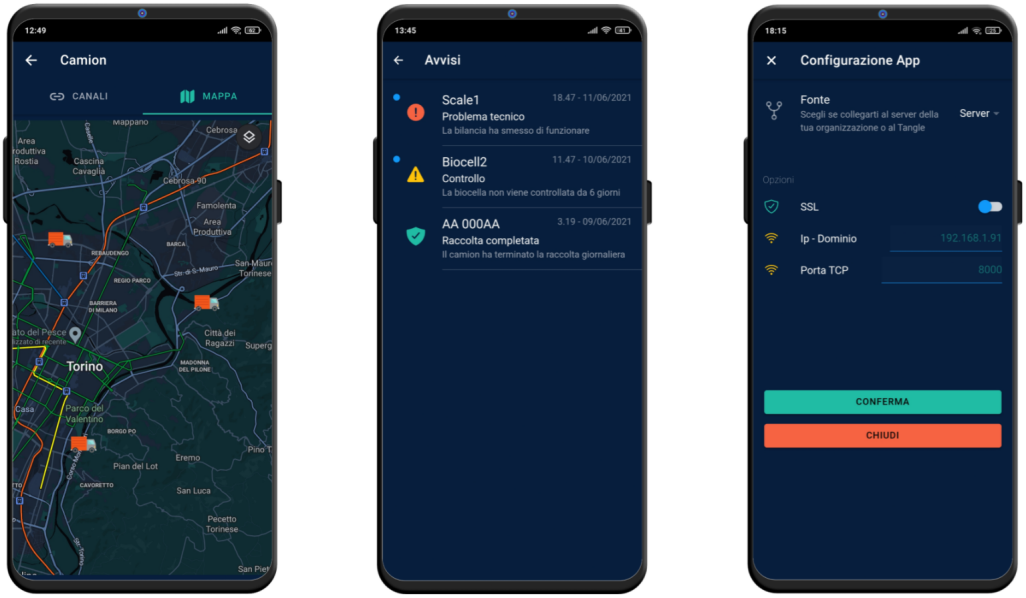

Figure 6.16: Views of extra further features and settings to switch from server mode to Tangle mode

Chapter 7

Conclusions

What makes an innovation so important is that it represents the umpteenth step towards the progress of human communities. However, creating new technologies is challenging and it is usually characterized by a huge level of uncertainty, risk and complexity. Creating a PoC is useful to test these technologies as it can determine whether an idea can be built in the real world and whether it is likely to be adopted by its intended users.

To help businesses to drive consistent ideas, the entire PoC process has been packed into five basic steps: from developing the idea, to firming it up, and presenting it to the investors.

The `Figure 7.1` reports and quickly explains what these steps are. Using this as a starting point, we can retrace all the steps done for the PoC of the BioEnPro4To project:

1. The first step of the PoC process is about **demonstration of the need of a product**. This would allow to determine if putting time and money into building some products, is supported by a proven necessity. In this case, the analyzed scenario reports the problems deriving from the current centralized infrastructures, that are susceptible to a lot of vulnerabilities and that lack of operational traceability and transparency. That is why it is needed to improve these infrastructures by enhancing the circular bioeconomy model supported by decentralized distributed ledgers. The technical idea to meet the challenge has been found in the use of IOTA.

2. After having defined the goals and the expectations, the next step, **concept development**, is characterized by an intensive phase of analysis during which as much information as possible about the idea must be collected. These information should ensure an overview of the actual possibilities such as the

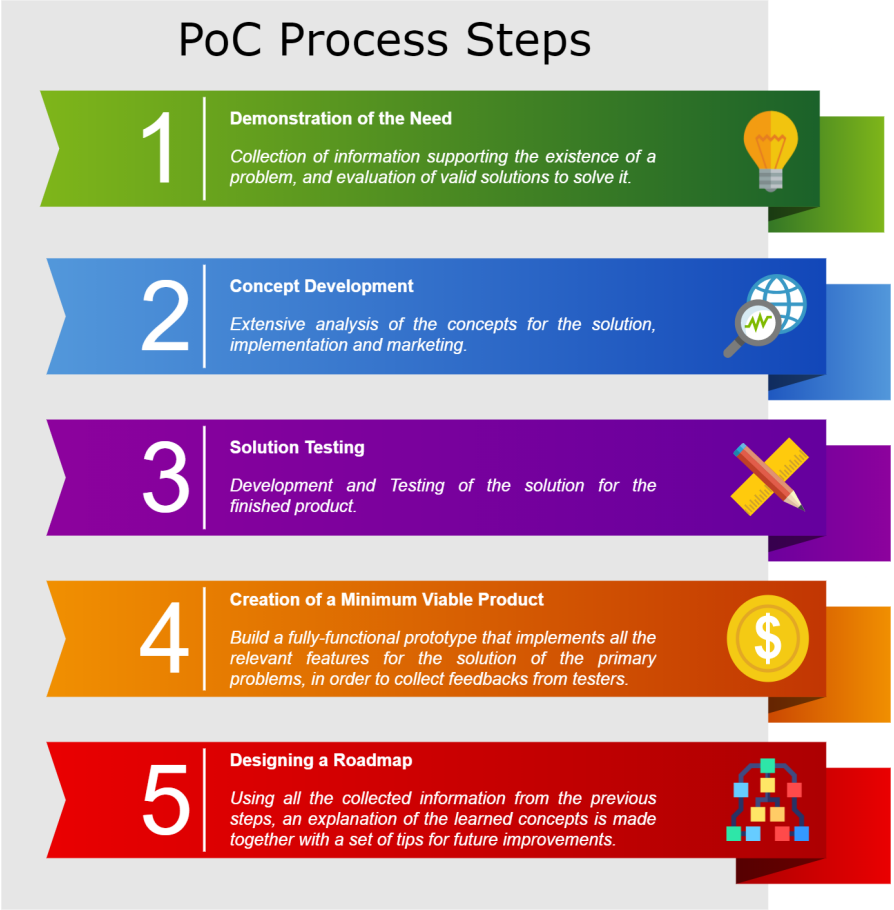

Figure 7.1: Representation of the PoC process steps.

existence of ready-to-use technologies or frameworks, and how these could solve the initial problems.

Before the actual implementation of this PoC, every benefit coming from the use of a circular bioeconomy model and their relative issues to enable them have been considered and analyzed. To solve these problems, IOTA has been chosen because of its features of decentralization, scalability and security together with its feeless and IoT oriented nature.

3. In this stage of **development and testing of the solution**, a prototype of the system is built.

   The PoC has integrated all the components provided by the IOTA ecosystem into a real case scenario. An architecture of the system has been designed,

implemented and tested in terms of performance and expected results, in order to make each component work in synergy.

4. In this phase, it is expected the **creation of a minimum viable product**. This should be a fully-functional solution that would include the most important features that are fundamentals for solving the primary problems. This step offers the opportunity to get more feedback to better understand the level of maturity of the prototype.

   Unfortunately, this stage has not been fully completed due to the lack of testers. This has not allowed us to collect enough feedback to understand the problems that might arise in production. However, some internal test results have already given us an idea of what are the core problems of the PoC. They will be discussed in the next sections.

5. The last stage is about **designing a roadmap** based on all the gathered information from the previous steps. An explanation of what have been learned during the development is made, together with a set of suggestion for further improvements.

   Everything that has been learned during the entire PoC process will be discussed and analyzed in the next paragraphs, as well as the future improvements the PoC might benefit from.

7.1 Obtained Results

The results, obtained by the final version of the entire system, have successfully proven that, the main problems the BioEnPro4To scenario is afflicted from, can be handled by the tools provided by IOTA. In particular we can affirm that:

- Operational transparency, traceability, security and trusted data provenance have been enabled by `Streams` and its (revisioned) `Channels` protocol.
- Centralized technologies, which are the ones the current waste management systems rely on, might be replaced by the IOTA based systems to enable decentralization and avoid single points of failure.
- Social awareness to stimulate more conscious behaviours among citizens would be possible thanks to the intrinsic properties of DLT. This will be essential to enable a circular bioeconomy model that strictly relies on the quality and on the variation of the materials the products are made of.

- Complex supply chain scenarios, involving a certain number of stakeholders, would become easier to handle thanks to systems like this one. As explained before, in the architecture design, future and possible stakeholder or actor integrations have been considered. This would allow to extend in a simpler and trustless manner the supply chain processes.
- Flexible authorization and authentication mechanisms for each actor are enabled through digital identities (`IOTA Identity`) allowing for a global cross-party solution.

To summarize, the final release of the PoC proposes an hybrid design of the architecture. A centralized part which is made of stakeholders managing internal servers. This is needed to provide a private environment for the actors belonging to a specific stakeholder, and to handle PII of the workers, that cannot be stored on the Tangle due to GDPR compliance. Moreover, each server (that is just one for the scenario analyzed in this PoC) is fundamental to maintain the correct channel tree structure, avoiding forks and data inconsistencies, and to provide the credential issuance service that enables the cross-party mechanism for actors authentication and authorization. On the other hand, a decentralized part made of the IOTA frameworks, `Streams` and `Identity`, that provide all the mechanisms needed to access the Tangle, and to exploit the protocols for channels and digital identities creation, enabling all the transparency, traceability and security benefits that this kind of technology offers.

Finally, the PoC includes a client, to support the *Edge Layer* and the actor interactions towards the server and the Tangle, and a mobile application capable of subscribing to the data streaming coming either from the server or the Tangle, in order to enhance internal monitoring operations.

7.1.1 Final Considerations

Despite all of these results, during the development of the PoC, I found the IOTA ecosystem not mature and reliable enough to permit a real deployment of the applications. `Streams`, in particular, which is the actual core component, is currently in the alpha stage, and it does not provide complete and flexible APIs that would be necessary, instead, for more complex use cases. The use of the channel protocol is quite tricky and limited, and the more complexity and flexibility are needed, the more limited the protocol becomes.

Although the use of a tree structure enables more flexibility in grouping and accessing daily channels, by using a hierarchical organization, it causes a drop in

performance, in terms of time needed for the correct tree management. This is due to the fact that more channels must be used in order to easily navigate from the root to the desired daily channel. Moreover, the sequential structure of a channel makes the server unable to serve in parallel requests that involve writing operations for the same portion of the tree. Otherwise, this could cause a fork into a certain channel that would break the entire navigation mechanism within the tree.

7.1.2 Performance Analysis

To better understand how the system performs during the publishing of data into a channel, some tests have been done. Thanks to these, we can understand how the size of the payload and the need for a specific throughput, in terms of packets per second, limit the performance of the system for the considered scenario.

The tests have been performed on the **IOTA Testnet**, by using a computer equipped with a 3rd generation Intel i7 mobile CPU. However, the performance are strictly related to the chosen node[1], that is (with the current version of `Streams`) the only responsible for the execution of the PoW, and it does not depend on the local CPU. This forces all the actors to maintain a stable internet connection, in order to keep appending messages to a channel. This causes another limitation, in fact, in an IoT scenario, where sensors may not be always in the same position (such as for a truck sensors), guaranteeing a stable internet connection may be a problem.

The tests involve the analysis of four different payload sizes, and for each of them 30 channels have been created. For every channel, 10 packets have been pushed consecutively, and each of them is made of a random sequence of characters split into public and masked parts according to the packet structure of our custom channel protocol. The sum of the sizes of these two parts is equivalent to the considered size. For instance, if the considered payload size is 1024B, each packet of each channel is made of a public part of 512B and a masked part of 512B.

The process that takes care of the creation of a channel involves the creation of the Announce message and, for each of the 10 messages, the generation of the random characters sequence, the encryption of the masked payload and the sending of the packet to the node. Then, the node will compute the PoW for each packet. An iteration of this process is indicated as `cycle`.

[1]Node URL: api.lb-0.h.chrysalis-devnet.iota.cafe

Table 7.1 and Figures 7.2 and 7.3 show the results obtained by these tests. As we can see, for small payload sizes (256B and 512B) 10 packets are sent in a reasonable amount of time (less than 9 seconds per packet) with a percentage of failure really close to zero. In these conditions, the built channel protocol seems to be reliable and fast enough for our use case. In fact, each actor would not probably have the need to send messages more frequently than this.

Things change for the last two payload sizes (1024B and 2048B): if the publishing time could be still reasonable for the throughput needed by an actor (less than 20.6 seconds per packet), the level of reliability of the protocol drops abruptly. The test for the first payload size, 1024B, managed to successfully complete 11 cycles over 30, and a total of sent packets which is 172 over 300. The second test, with 2048B payload size, have completed just 4 cycles and 136 sent packets. Each failure in these tests comes from the fact that the response coming from the node is long in coming, and the request timeout expires. This behaviour affects the correct functioning of a channel, because if the author does not notice that the packet has been published despite the expiration of the timeout, there is the chance to create a fork within the channel that would break the entire mechanism for the reading process. However, this can be solved by introducing a mechanism that checks the actual state of the channel after each expiration of timeout, so that the author's internal state is updated and consistent. But the cost would be introducing one more mechanism that would slow down the process.

Payload Size	**Sent Packets in 30 Cycles**	**Cycles Success Rate**	**Cycle Avg Publishing Time**
256 B	300/300	30/30	**83.07s**
512 B	297/300	29/30	**86.34s**
1024 B	172/300	11/30	**186.82s**
2048 B	136/300	4/30	**205.12s**

Table 7.1: Test results for a set of four different payload sizes. The average time values have been computed by using a number of samples equal to the number of cycles that have successfully ended.

More in detail, we can also notice the cycle publishing time is not constant. Figures 7.2 reports also the minimum and the maximum value recorded in the 30 cycles process and, As we can see, the black vertical line in the middle of each bar shows how big is the difference between the maximum and the minimum value. This means that the result in a certain moment is likely to be related to the

workload of the node as well as the network traffic conditions. This hypothesis is also supported by `Figures 7.3` that shows how the different trends are not regular and it is not possible to clearly identify or predict them.

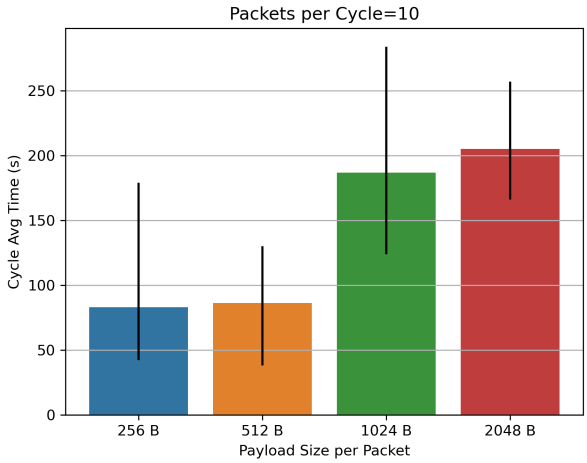

Figure 7.2: Minimum, Maximum and Average value of cycles publishing time for each payload size.

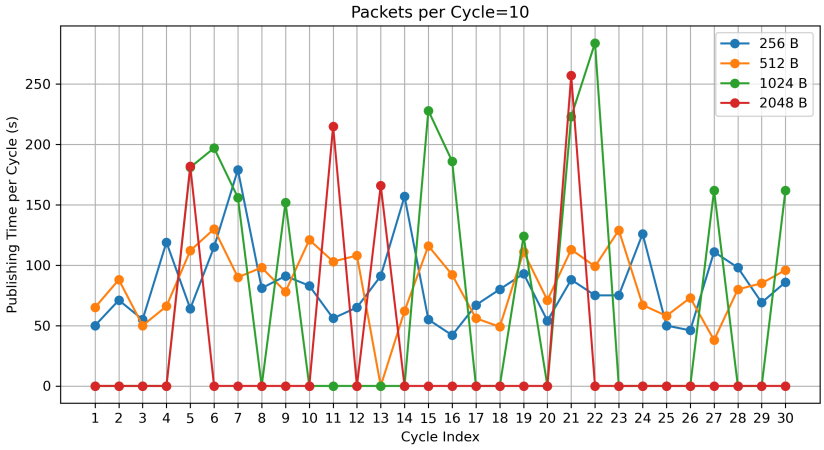

Figure 7.3: Cycles publishing times trend for each payload size. The zero values refer to cycles that have been interrupted due to timeout expiration.

7.2 PoC Setup

In order to correctly configure and run the system, instructions are reported hereafter together with a brief explanation of the configuration parameters.

To setup either the server and the client, a `.env` file is provided containing some configuration variables.

7.2.1 Server Configuration

1. The first thing to do is to clone the repository by using the related git command:

```
git clone https://github.com/lore-lml/bioenpro4to_gateway.git
```

2. If needed, in order to customize the initial data of the tables, the insert statements of the `db_docker_init/sql/init_db.sql` file can be edited.

3. Then, it is possible to set up the configuration variables of the server by editing the `.env` file. The `Tables 7.2` shows the most important configuration variables with the relative description of what they need for.

4. Finally, the application can be put into a container using docker by simply using the command:

```
docker-compose up -d
```

Configuration Variable	Description	Default behaviour if not set
`SERVER.ADDR`	Ip address of the server. `127.0.0.1` for localhost or `0.0.0.0` to make it available to the whole network.	`127.0.0.1`
`SERVER.PORT`	TCP port of the server.	`8080`
`IOTA.MAINNET`	`True` to select the IOTA mainnet or `False` for testnet	`False`
`IOTA.ROOT_CHANNEL.ADDR`	Address of an existing root channel. If it is set it tries to import the tree with the corresponding root address.	It creates a new tree
`IOTA.ROOT_CHANNEL.PSW`	Password for the encryption of the `State` message of each channel of the tree. Daily channels are not included.	A default password is provided
`IOTA.IDENTITY .STORAGE.TYPE`	`Stronghold` to persist the data about the digital identity of the `Issuer`. Otherwise it is temporarily stored on RAM.	Temporary storage
`IOTA.IDENTITY .STORAGE.PSW`	Password to access the identity vault.	A default password is provided
`IOTA.IDENTITY .ISSUER_NAME`	Name of the `Issuer`. This field must always be provided.	`None`

Table 7.2: Server configuration variables.

7.2.2 Client Configuration

The client library can be used just on **Linux machines**.

1. As before, clone the github repository by using:

```
git clone https://github.com/lore-lml/bioenpro4to_http_client.git
```

2. Edit the `.env` file to set up the configuration variables. `Tables 7.3` reports the list of the most important ones.

3. Use `make build` or `make run` bash commands to build or run the example application respectively.

Configuration Variable	Description
`HOST.ADDR`	Ip address of the server.
`HOST.PORT`	TCP port of the server.
`IOTA.IDENTITY.STORAGE.TYPE`	`Stronghold` to persist the data about the digital identity of the `Holder`. Otherwise it is temporarily stored on RAM.
`IOTA.IDENTITY.STORAGE.PSW`	Password to access the identity/credential vault.
`ACTOR.AUTH_PSW`	Password to provide to be authenticated by the server when a credential is requested.
`ACTOR.CHANNEL_PSW`	Password used to encrypt the `State` message of the daily channels of the actor.
`ACTOR.ID`	Id of the actor used as username to authenticate to the server

Table 7.3: Client configuration variables.

7.3 Open Issues and Further Improvements

Although the results of this early implementation of the PoC seem to be promising, the used technologies must be consolidated yet. In particular, the lack of a complete and stable release of a **selective permanode**[2] or an environment to write reliable smart contracts is a big barrier to reach an hypothetical market-ready production stage. These are indeed the main problems to solve once the platform will become more mature: the first because data are currently temporarily available on the Tangle until a snapshot is performed, while the second because smart contracts would be essential tools to record interactions among stakeholders in a trustless manner. At the current stage of the PoC, in fact, stakeholders should take care personally of tracing each interaction that involves one of their actors, and this means that, without a neutral intermediary, one of the two sides must trust the other one that is in charge of publishing the interactions history.

This is just the starting point to turn this PoC into a system for a real use case. I have reported a few more tips, ordered by priority in `Table 7.4`, to improve the prototype performance and the developers experience. Both private and public companies or institutions that want to implement and to test a system similar to the one reported in this thesis, are encouraged to consider these suggestions as well as checking for future updates of the IOTA ecosystem that might enable new interesting features for such cases.

[2]Currently the official IOTA permanode solution, `Chronicle`, must store the entire transaction history including all the zero-value-transactions. This enables persistence for applications that use channels to publish some data to the Tangle, but at the cost of storing also unnecessary data causing potential storage size issues. For this reason, a new release of `Chronicle` is expected to come in order to allow the permanode to select and store just the data of interest, according to some user-defined filters. This new feature is still under development.

Priority	Tip Description
1	**Enabling persistence** using the future release of `Chronicle` for a selective permanode solution.
2	**Integration of smart contracts** for an easier and improved management of stakeholders interactions. Despite IOTA being designed with a feeless transaction system, the introduction of smart contracts would enable new business opportunities for companies or people in charge of running the code. In fact they might ask for some kinds of rewards in exchange for the execution of the smart contract.
3	**(i)** Instead of using a centralized server for the roles of channel manager and credential issuer, the business logic to authenticate and authorize a certain author together with the channel management process, it would be a great improvement to use a smart contract as the only responsible for this. It would achieve an **entirely decentralized system** that would make actors even more independent from the stakeholder they belong to. **(ii)** Otherwise, if the solution that involves the use of a server for each stakeholder is kept, the **creation of a cluster** (for instance using Docker containers) is encouraged. In this case, each node of the cluster would take care of just a portion of the tree channel structure, guaranteeing a **greater throughput** in terms of parallel execution. For example, considering the same scenario of the PoC, each node might have been responsible for just a single category among the proposed ones.
4	**Tuning of the granularization** of the actor channels. Instead of using daily channels, it can be considered to use a greater or lower granularity according to the use cases. For example, each actor may create channels every hour instead of every day. This guarantees an improvement of performance per single channel but a higher complexity of the tree structure.
5	Improving the UI of the mobile application for a more efficient and comfortable experience on bigger devices such as tablets.

Table 7.4: Suggestions for further improvements of *BioEnPro4To-like* projects for interested companies and institutions.

List of Figures

List of Tables

Acronyms

DLT
Distributed Ledger Technology

DDBMS
Distributed Database Management System

BFT
Byzantine Fault Tolerance

PoW
Proof of Work

PoS
Proof of Stake

ECDSA
Elliptic Curve Digital Signature Algorithm

EdDSA
Edwards-curve Digital Signature Algorithm

NFT
Non Fungible Token

P2P
Peer-to-Peer

IoT
Internet Of Things

M2M

Machine-to-Machine

DAG

Directed Acyclic Graph

OTS

One Time Signature

WRW

Weighted Random Walk

PII

Personally Identifiable Information

PoC

Proof of Concept

Bibliography

[1] *Distributed Ledger Technology: beyond block chain (PDF Report)*. Jan. 2016. URL: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf (cit. on p. 1).

[2] *Blockchain - Wikipedia*. URL: https://en.wikipedia.org/wiki/Blockchain (cit. on pp. 4, 5).

[3] Nirupama Devi Bhaskar; David LEE Kuo Chuen. «Bitcoin Mining Technology». In: (2015), pp. 45–65 (cit. on p. 6).

[4] Veronica Valsecchi. *La classificazione delle Blockchain: pubbliche, autorizzate e private*. Dec. 2018. URL: https://www.spindox.it/it/blog/la-classificazione-delle-blockchain/ (cit. on p. 7).

[5] *Proof of Authority - Wikipedia*. URL: https://en.wikipedia.org/wiki/Proof_of_authority (cit. on p. 7).

[6] Diego Geroni. *Hybrid Blockchain: The Best Of Both Worlds*. Jan. 2021. URL: https://101blockchains.com/hybrid-blockchain/ (cit. on p. 7).

[7] *La Byzantine Fault Tolerance spiegata*. Jan. 2021. URL: https://academy.binance.com/it/articles/byzantine-fault-tolerance-explained (cit. on p. 8).

[8] *Proof of Work - Wikipedia*. URL: https://en.wikipedia.org/wiki/Proof_of_work (cit. on p. 9).

[9] Giulia Spinoglio. *Proof of Stake, cos'è, perché sta soppiantando il Proof of Work*. Apr. 2021. URL: https://www.blockchain4innovation.it/esperti/proof-of-stake-cose-perche-sta-soppiantando-il-proof-of-work/ (cit. on p. 12).

[10] *A short history of cryptocurrencies*. URL: https://daviescoin.io/blog/a-short-history-of-cryptocurrencies.html (cit. on p. 14).

[11] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. URL: https://bitcoin.org/bitcoin.pdf (cit. on p. 15).

[12] *Hashcash - Bitcoin Wiki*. URL: https://en.bitcoin.it/wiki/Hashcash (cit. on p. 19).

[13] *Proof of Work - Bitcoin Wiki*. URL: https://en.bitcoin.it/wiki/Proof_of_work (cit. on p. 19).

[14] *Weaknesses - Bitcoin Wiki*. URL: https://en.bitcoin.it/wiki/Weaknesses (cit. on pp. 26, 28, 29).

[15] Raffaele Battaglini. *Smart Contracts in Breve*. URL: https://www.4clegal.com/hot-topic/smart-contracts-breve (cit. on p. 31).

[16] *What is Tokenization*. URL: https://etorox.com/what-is-tokenization/#:~:text=Tokenization%5C%20is%5C%20a%5C%20process%5C%20where,or%5C%20recorded%5C%20on%5C%20a%5C%20blockchain.&text=Tokenization%5C%20in%5C%20simple%5C%20terms%5C%20converts,manipulated%5C%20on%5C%20a%5C%20blockchain%5C%20system. (cit. on p. 34).

[17] *Biobased economy - Wikipedia*. URL: https://en.wikipedia.org/wiki/Biobased_economy (cit. on p. 35).

[18] Raja Wasim Ahmad; Khaled Salah; Raja JayaramanRaja Jayaraman; Ibrar YaqoobIbrar Yaqoob; Mohammed Omar. *Blockchain for Waste Management in Smart Cities: A Survey*. Apr. 2021. URL: https://www.techrxiv.org/articles/preprint/Blockchain_for_Waste_Management_in_Smart_Cities_A_Survey/14345534/1 (cit. on p. 37).

[19] Bahar Farahani; Farshad Firouzi; Markus Luecking. *The convergence of IoT and distributed ledger technologies (DLT): Opportunities, challenges, and solutions*. Dec. 2020. URL: https://www.sciencedirect.com/science/article/abs/pii/S1084804520303945 (cit. on p. 43).

[20] *HORNET - The IOTA community node*. URL: https://github.com/gohornet/hornet/blob/main/README.md (cit. on p. 47).

[21] *Ternary system – What is it and is there an advantage?* URL: https://iota-beginners-guide.com/future-of-iota/iota-x-0-ternary-vision-abandoned/ternary-systems/ (cit. on p. 50).

[22] *The Tangle: an illustrated introduction*. Feb. 2018. URL: https://blog.iota.org/the-tangle-an-illustrated-introduction-f359b8b2ec80/ (cit. on pp. 55–57).

[23] Luca Grande. *Conservazione dei dati alla luce del recente Provvedimento della Banca d'Italia*. June 2020. URL: https://www.antiriciclaggiocompliance.it/conservazione-dei-dati-alla-luce-del-recente-provvedimento-della-banca-ditalia/ (cit. on p. 58).

[24] *Streams Specification - Rev: 1.0 A*. URL: https://github.com/iotaledger/streams/blob/develop/specification/Streams_Specification_1_0A.pdf (cit. on p. 68).

[25] Jelle Femmo Millenaar; Matthe Yarger. *The Case for a Unified Identity*. URL: https://files.iota.org/comms/IOTA_The_Case_for_a_Unified_Identity.pdf (cit. on pp. 78, 81, 82).