

POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

A decentralized marketplace for m2m economy within smart cities

Relatore

Prof. Guido Perboli

Alessandro Manfredi

Marzo 2019

*A chi ha sempre creduto in me
e continuerà a farlo.*

Acknowledgments

English

First of all, I would like to thank my parents, Ornella and Thierry, for giving me the opportunity to reach this goal and for making me the person I am today. I greatly thank Valentina for supporting me giving me so much love and affection during this last part of my study. A big thank you also goes to my uncles Nelsa and Luciano and my grandmother Maria, who have always supported me with so much love. I also thank all the rest of my family: my grandmother Anna, my uncles Leo and Germana and my cousins Michele, Andrea and Sara. Another great thank you goes to all my dearest friends Niki, Lollo, Tortug, Box, Menego, Ciuni, Bomber, Ieie, Pagan, Fede, Marco, Diddi, Giorgia and Viola. I also thank Professor Guido Perboli for supporting me in this thesis.

Italiano

Innanzitutto vorrei ringraziare principalmente i miei genitori Ornella e Thierry, per avermi dato la possibilità di raggiungere questo traguardo e per avermi reso la persona che sono oggi. Ringrazio enormemente Valentina per avermi supportato donandomi tantissimo amore e affetto durante questa ultima parte del mio percorso di studio. Un forte ringraziamento va anche ai miei zii Nelsa e Luciano e alla mia nonna Maria, i quali mi hanno sempre supportato con tanto amore. Ringrazio inoltre tutto il resto della mia famiglia: mia nonna Anna, i miei zii Leo e Germana e i miei cugini Michele, Andrea e Sara. Un altro grandissimo ringraziamento va a tutti i miei più cari amici Niki, Lollo, Tortug, Box, Menego, Ciuni, Bomber, Ieie, Pagan, Fede, Marco, Diddi, Giorgia e Viola. Ringrazio molto anche il professore Guido Perboli per avermi supportato al meglio durante questo lavoro di tesi.

Contents

Acknowledgments	ii
Abstract	1
1 Introduction	3
1.1 Blockchain and its story	3
1.2 IOT (Internet Of Things)	6
1.3 Smart Cities	6
2 Blockchain nowadays	1
2.1 The Basics of Bitcoin	1
2.2 Current problems of the Blockchain	6
2.3 2th-generation Blockchain technology	7
2.4 3th-generation Blockchain technology	8
2.5 IOTA	11
2.5.1 Tangle	13
2.5.2 Basic Concepts	17
2.5.3 MAM: Masked Authentication Message	20
2.5.4 Qubic: Quorum-based Computation	22
2.6 IoT and Blockchain: applications and problems	26
3 Development	33
3.1 Architecture	37
3.2 Functionalities	38
3.3 Marketplace	56

4 Computational results 66

4.1 Receive transaction 68

4.2 Send transaction 69

4.3 Buy data stream from sensor 70

5 Conclusions 74

Abstract

As global urbanization continues, resources will begin to run low. Health care will not be able to keep up. Energy demands will exceed current capacity. Where will everyone work? How can governments and policy makers facilitate an environment conducive to economic growth? Cities around the world are working to become what we call "smart cities", an environment for addressing the challenges of advanced urban planning, energy and transportation management and urbanization by combining new technology. They're using IoT connected devices to do everything from detecting gunshots to monitoring traffic and air quality. According to a report by McKinsey, by 2020 the number of smart cities will be about 600 worldwide, and 5 years later almost 60 percent of the world's GDP will be generated by them. Digital technologies could be the pivot of economic progress, and Blockchain could be one of them. Since there are no standards and the requirements are not equal in each cities, the technology infrastructure is left up to each of them. That is where Blockchain can be used. Blockchain and Distributed Ledger are able to connect these technologies together. The more technologies we connect using this kind of technologies, the more value can be drawn. Imagine that a city has a digital ledger in which every house has a presence containing all relevant information about itself, from property ownership, mortgage balance and to transactional data like utility use, past and current contractor relationships, and property tax assessment. The city is able to access these in order to better perform administrative tasks related to the property and coordinate services with greater accuracy. The property owner would have a verified, reliable way to perform transactions like hiring contractors to do lawn work, renting rooms, or selling power generated by solar panels installed in the property. Now imagine extending that to the city's broader infrastructure. A traveler take a autonomous electric bus at the station in order to reach the city center. Knowing from traffic sensor data that there's a traffic, the bus automatically chooses an alternate path. While the traveler is traveling, thanks to his phone he is able to directly pay

the company that offers the transport service in real time. Since the traveler is connected to the city's public wifi, s/he immediately receives a notification. For this reason the need arises to create an environment in which end users are able to monitor and interact with devices in the city as it may be useful for a citizen to monitor some of these data. The aim of this thesis work is to develop a system that allows the user to purchase and monitor data streams coming from the smart city thanks to the use of IOTA technology.

Chapter 1

Introduction

1.1 Blockchain and its story

Blockchain technology is one of the most important invention of the 21st century. It is a ledger of facts, replicated across computers belonging to a peer-to-peer network. Facts can be content signature or monetary transactions and members of the network are called nodes. All communication inside the network takes advantage of cryptography to securely identify the sender and the receiver. When a node wants to add a fact to the ledger, a consensus forms in the network to determine where this fact should appear in the ledger; this consensus is called a block. When a node wants to generate a new fact, before being written on this register, it must be approved by the whole network. To reach this goal a technique called consensus is used. The result of this consensus mechanism is called block. The most known application of the Blockchain technology is Bitcoin, a distributed network that maintains a ledger of balances of a cryptocurrency called Bitcoin. The system allows payments to be sent between users without using a central authority, such as a bank. Bitcoins aren't printed, like dollars or euros, they are produced by computers all around the world. This was the first example of what we call cryptocurrency, an asset class that shares some characteristics of traditional currencies, with verification based on cryptography. In 2008, through a mailing list, a person called Satoshi Nakamoto sent a paper called "Bitcoin: A Peer-to-Peer Electronic Cash System". After more than ten years the speculation of this name has never ceased to exist as so many people have declared themselves to be the undersigned. Still, 10 years later it is unknown who the creator of Bitcoin is. Although no one has yet been able to give a name to the creator of Bitcoin

whitepaper, there are many theories that link mathematicians and cryptographers to that name to some people. Much research has suggested that the creator or creators of Bitcoin can hide behind the name of 5 people. The first one is Nick Szabo who was involved in developing "Bit Gold", a digital currency that predates Bitcoin, and for this reason, the decentralized currency enthusiast is thought to have participated in the creation of Bitcoin. Bit Gold is thought to have been one of the earliest predecessors to Bitcoin, thanks to the fact that network participants would consume a lot of computing power to solve cryptographic puzzles where the solution would become part of the next challenge.

Another case is that of the Californian resident Dorian Prentice Satoshi Nakamoto who has been thought to have been the author behind the Bitcoin white paper. He has been contacted by a journalist to talk about Bitcoin and he responded "I am no longer involved in that and I cannot discuss it. It's been turned over to other people. They are in charge of it now. I no longer have any connection." Another person is Hal Finney, a computer scientist and a cryptographer who as a young man was involved in the cyberpunk movement and was very active on online forums. In addition, Finney was the first person to receive a bitcoin payment from Satoshi, which is why it is believed that he himself generated this payment. Born in October of 1970, Craig Wright has defined himself to be one of the creators of bitcoins even if there are no matches that would validate this statement. The last one is Dave Kleiman who has a pending lawsuit against Craig Wright describes that both men reportedly worked together on the Bitcoin developing process. Court documents allege the two men collaborated to develop the Bitcoin network. Following its launch, the two later supposedly sent each other Bitcoin transactions which were recorded on the Blockchain.

Court documents the two men collaborated from March of 2008 to develop the Bitcoin network. Following the launch of Bitcoin, they sent each other Bitcoin transactions which were recorded on the Blockchain. Many companies use this word to mean some sort of magical device by which all their data will never be wrong. Basically, a Blockchain is a linked list of blocks, which contain a group of ordered transaction. It implements a sort of new generation database. The main thing distinguishing a Blockchain from a normal database is that there are specific rules about how to put data into the database. That is, it cannot conflict with other data that is already in the database (consistent), it can only be inserted and not removed (immutable), and the data itself and the data itself is associated with an owner (ownable), it's available and

replicable. Finally, everyone agrees on the state of the data within the database (canonical) without a central party (decentralized). The immutable audit trail uncontrolled by any central party is certainly useful, but there are many costs to create such a system. Let's examine some of the issues. A small bug could corrupt the entire ledger or cause some conflicts. Of course, a corrupt ledger can not guarantee any kind of data consistency. Furthermore, all such systems have to be designed and implemented in order to be consistent. In case of breakages things, you generate conflicts and the ledger becomes corrupted. In a normal database in case of inconsistency it would only be enough just to fix the it, but in this kind of system it is not enough: it is necessary the agreement of all players (consensus) in the system, in order to change the ledger. Another issue is that the maintenance is very costly. In a traditional database to write, read or transmit the value of a data it is necessary to write it, read it and transmit it only once, while in a blockchain it is necessary to perform these operations thousands of times. Decentralized networks aren't new. For example Napster and BitTorrent are P2P networks. Instead of exchanging music or movies, members of the Blockchain exchange facts. The biggest problem in distributed systems is the resolution of conflicts generated by the network. Relational databases offer referential integrity, but there is no such thing in distributed system. If two incompatible facts arrive at the same time, the system must be able to determine which of these facts is considered valid. In a P2P network, two facts sent at the same time may arrive in different orders. To guarantee integrity over a P2P network, it is needed a way to make everyone agree on the ordering of facts. It is necessary a consensus mechanism. Blocks are a way to handle facts in a network of non-trusted peers. Each block contains an ordered sequence of facts, and there is only a single chain of blocks, replicated in the whole network. These facts are inserted into a block thanks to a process called Mining. Each block is identified by using a cryptographic hash (also known as a digital fingerprint). The block generated will contain the hash of the previous block, so that blocks can form a chain from the first block ever (known as the Genesis Block) to the newest block. In this way, all the data are be connected via a linked list structure.

1.2 IOT (Internet Of Things)

Internet of Things is the concept of inter-networking of physical devices, embedded with electronics, software, sensors, actuators, and network connectivity that enable these objects to collect and exchange data. A "thing" in the internet of things can be a sensor installed in the ground next to a plant that allows to detect the diametral growth of it, a car that allows to notify the driver in case of vehicle breakdown or any other natural or any object to which an IP address can be assigned so that data can be transferred over the network. The term Internet of Things is 16 years old. The word "Internet of Things" was invented by Kevin Ashton in 1999 during his work at ProcterGamble. He who was working in supply chain optimization and he wanted to get attention from senior management so he could explain the new technology called RFID. Since Internet was the leading topic of that era, and since the topic he was dealing with made sense, he decided to call it "Internet of Things". An IoT ecosystem is nothing more than an environment of devices connected to the web equipped with embedded processors, sensors and communication hardware. Moreover, these devices are able to collect, send and receive data. IoT devices share the data they collect by sending them to an IoT gateway or other edge device where these data is either analyzed locally or sent to the cloud to be analyzed. These devices are able to operate independently without human intervention although people can interact with them, for instance to set them up, access the data or to give them instructions. The current number of internet connected "things" already exceeded our population back in 2008. By 2020 this number is expected to reach 50 billion. IoT smart devices are mainly used in manufacturing, healthcare, business and in our homes or phones. By 2025 the global worth of IoT tech is projected at 6.2 trillion of dollar.

1.3 Smart Cities

A smart city combines information, communication technology, and the network of physical devices referred to as the Internet of things to make city operations, commuting, quality of life, and sustainability as optimal as can be. A smart cities can be subdivided in 3 main layer. Smart cities are urbanized areas that use information technology and sensors that collect data to communicate with the people, devices, and structures that populate it. This data are processed

and analyzed for example for hospitals, libraries, traffic monitoring and transportation systems, water supply networks, power plants, waste management, information systems, law enforcement, and other services. Smart cities also allow citizens to monitor and interact in real time the evolution of it. Information technology is used to reach interactivity, quality and performance of urban services, to reduce costs and resource consumption and to increase contact between citizens and government. Ideally a smart city should: increase connectivity, make business work more efficient, contribute to improving the quality of life of citizens living in it and prevent disasters before they happen. Some examples of smart solutions that a city could implement would be:

- **Smart Mobility and Transportation:** traffic management and congestion within the city is made by smart roads that prevent these events from occurring often. They should also be able to react to accidents in an efficient manner and reduce delays as much as possible.
- **Smart Energy Systems:** Real time management of plumbing systems allow the citizen to monitor, for example, the percentage of bacteria inside the water and understand if it is clean. Furthermore these systems can be used for irrigation purposes, and for plugging leaks before they can have a profuse affect.
- **Smart Waste Management and Recycling Planning:** Sensors put in waste containers would allow a much more efficient waste disposal, in this way also the recycling process would be improved.
- **Smart Pollution Containment:** Sensors placed within the city actively read and report air pollution, CO2 levels ,toxic waste in the ground, humidity, as well as impending storms or natural disasters. Cities are kept healthy and clean, and the citizens are aware of the affect the outside environment will have on their health.
- **Smart Safety:** Drones, facial recognition, street lights would allow the police greater control in the city so as to reduce crime, prevent violence and increase security in cities

Chapter 2

Blockchain nowadays

2.1 The Basics of Bitcoin

As it is mentioned in [1], Blockchain technology allows value exchange without the need of central and trusted part. The most known application of the Blockchain technology is Bitcoin, a digital currency that is used to exchange products and services. To keep track of the amount of Bitcoin each of us owns the Blockchain hystory. **ledger** (Fig. 2.1).

LEDGER	
Account owner	Value
Mary	4
John	56
Sandra	83
Lisa	16
David	187
Brian	23
...	...

Figure 2.1. Bitcoin ledger digital file simplified

The ledger file is not maintained on a central server, but is distributed through private (and non) computers that maintain the (updated) status of it and execute operations on it. Each of these computers represents a **node** of the Blockchain network, which possess a copy of the ledger. for that reason it is possible to define the Blockchain as a system that allows a group of connected computers(nodes) to keep a single updated ledger. In order to perform transactions on the Blockchain, a user need a wallet, a program that allows him to store and exchange his

keys which needs to access his Bitcoins. In order to guarantee that Bitcoins of a user are spent only by the user itself, each wallet, thanks to cryptographic method uses a unique pair of distinct but related keys: a private and a public key. If a message is encrypted with the public key, only the person with the private key relating to the public key used for encryption is able to decrypt the message. If the message was encrypted with the private key, all the public key holders would be able to decrypt the contents of the message, For example: when a person wants to send Bitcoins, he needs to broadcast a message encrypted with his private key related to his wallet. Since this person is the only one who knows the private key, he is the only one who can spend his Bitcoins. Each node in the network can cross-check that the transaction request is coming from David by decrypting the message with the public key. When a user encrypts a transaction, he uses his private key in order to generate signature that will be used by each nodes belonging to that network, to verify the source and authenticity of the transaction. The digital signature is a string of text resulting from transaction encrypted with the private keys. If a user change only a single character in the transaction message, the digital signature will change, so no potential attacker can change his transaction requests or alter the amount of Bitcoin he is sending (Fig. 2.4).

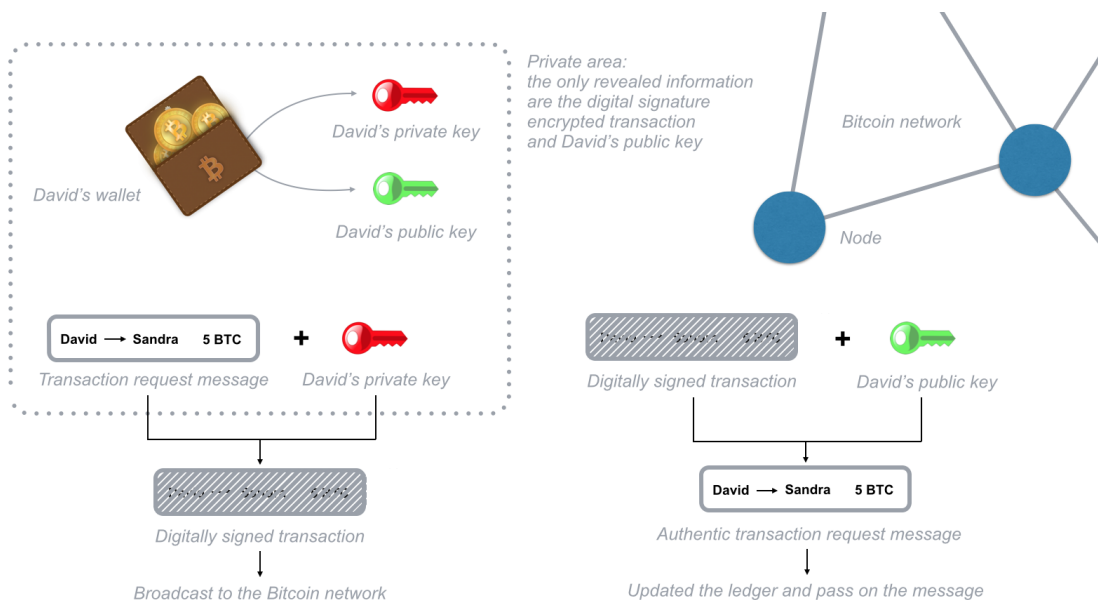


Figure 2.2. Transaction request message simplified

To send Bitcoin a user needs to prove that he owns the private key of a specific wallet as he need the key to encrypt his transaction request message. Since he broadcasts the message only

after it has been encrypted, he never has to reveal his private key. Each node in the Blockchain keeps a copy of the ledger therefore to determine a wallet balance, it need to analyze and verify all the transactions that ever took place on the entire network connected to a wallet since the Blockchain doesn't keep track of all balances in the network, it keeps track of all transactions related to each address. For that reason it is possible to calculate the balance (Fig. 2.3).

LEDGER	
Transactions	Value
Mary → John	10.000
John → Lisa	0.345
Sandra → David	18.4332
Lisa → Sandra	7.156
David → Mary	12.3402
Brian → Lisa	3.029381
...	...

Figure 2.3. Blockchain Ledger simplified

The balance verification is performed by checking all previous transaction. For example: Alice wants to send 10 Bitcoins to Bob. In order to send 5 Bitcoins to him, Alice has to generate a transaction that includes all previous inputs transactions which must guarantee at least 5 Bitcoins. Once the transaction has been generated, all the nodes in the network check that Alice has 5 Bitcoin by checking all previous inputs. This is all performed automatically in Alice's wallet and double-checked by the Bitcoin network nodes; she only sends a 5 Bitcoin transaction to Bob's wallet using his public key (Fig. 2.4).

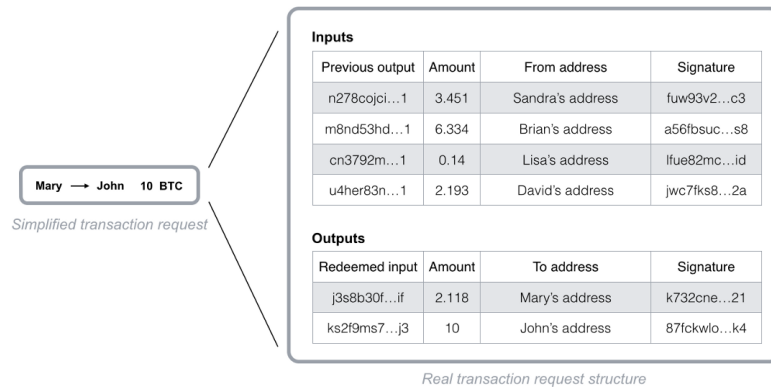


Figure 2.4. Blockchain transaction request structure

In order to check if input transactions are valid, the system checks all the previous transactions correlated to a wallet. To speed up the verification process, a special record of unspent transactions is kept by the network nodes, so as not to allow double spending operations. Owning bitcoin means that on the ledger there are incoming transactions that point to an address and that these inputs have not yet been spent. Anyone can anonymously access the Bitcoin network, for example through a VPN or using TOR and submit or receive transactions revealing nothing more than his public key. However if someone uses the same public key for several times, it's possible to connect all the transactions to the same owner. For that reason it is said that Bitcoin is pseudo-anonymous. The Bitcoin network allows to generate as many pair of private and public key as an user likes.. This allows to receive payments on different wallets, and there is no way for anyone to know that a user owns all these wallets' private keys, unless he sends all the received Bitcoins to a wallet. The total number of possible Bitcoin addresses is:

$$2^{160} = 1461501637330902918203684832716283019655932542976 \quad (2.1)$$

Bitcoin groups the transactions in time order within a **blocks**, each of which contains a link to the previous block. This link between the blocks forms the structure that gives the name to the system: **Blockchain** (Fig. 2.5) .

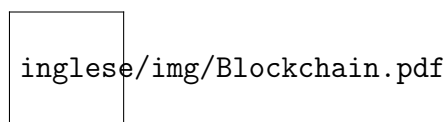


Figure 2.5. The block chain sequence structure simplified

Transactions within a block can not be changed and are considered validated while those not yet in a block are considered unconfirmed . Each node can group together transactions into a block and broadcast it to the network as a suggestion for which block should be next. In order to be added to the Blockchain, each block must contain the solution to a very complex mathematical problem based on the hash function. The only way to solve such a mathematical problem is to choose a random numbers that, combined with the previous block content, generate a defined result. The choice of this number continues until the problem is solved. The node that solves the mathematical problem gets the right to place the next block on the chain and broadcast it to the whole network. It is possible that two nodes solve this problem at the same time and send their blocks to the network simultaneously, both blocks are broadcast and each

node builds on the block that it received first. However the Blockchain system requires each node to build on the longest chain available. So in case of ambiguity about which is the last block, as soon as the next block is solved, each node will use the longest chain as the only option (Fig. ??). A reward is given to the users who solve the mathematical problem of each block. The activity of running the Bitcoin Blockchain, in order to obtain these Bitcoin rewards is called **mining** and it's similar to the process of mining gold. Rewards are the main incentive for people to run the nodes, thus providing the necessary computing power to process transactions.

In order to send Bitcoins, user needs to have at least reference within an incoming transaction to his owns wallet. A cryptocurrency wallet is basically a software that takes care of generating a user's public and private key pair. The task of these wallets is to facilitate the operations of sending, receiving and monitoring the balance of a user To be able to spend those coins and unlock the funds, the private key stored in your wallet must match the public address the currency is assigned to. It is important to understand that the user does not have its cryptocurrencies in his wallet, but he owns the keys with which it is possible to access its cryptocurrencies on the Blockchain. There are two types of wallet that differ depending on if the keys are generated and related to each other, these are **deterministic** and **non-deterministic wallets**.

- **non-deterministic wallet:** This type of wallet is initialized and during this process a series of keys are generated and also allow the generation of additional keys when requested. This is a point in favor of security in fact, none of the keys having a relationship to each other. Since there is no relationship between these keys, it is necessary to save them independently.
- **deterministic wallet:** This more modern type of wallet allows generating a series of keys by applying a hash function to a seed. In this case it is necessary to save only the seed because for example in case of loss of the wallet it is only necessary to restore it with the same seed using previously. This allows the user to resote the wallet quickly. Seeds are typically serialized into human-readable words.

2.2 Current problems of the Blockchain

The use of Blockchain will grow exponentially with the passing of the years, nowadays, however, presents some problems:

- **Environmental cost:** Blockchain is based on encryption mechanism to guarantee security and generate a consensus on a distributed network. The Blockchain is based on complex cryptographic functions used to reach consensus, which require great computational skills. Currently, in order to reach consensus within the system, the proof of work algorithm is used, which, as previously mentioned, is based on the resolution of a very complex mathematical problem. To solve it relies on very powerful video cards that allow you to solve this algorithm in the shortest time possible. The problem of such video cards is the cost of operation which consume so much electricity. Last year it was found that the computing power required to keep the bitcoin network operational consumed energy equal to that consumed by 159 nations.
- **Lack of regulation creates a risky environment:** Although the term ICO would seem an innovative and revolutionary concept, as this technique would allow to receive funding without too many hitches, the regulation of this is still not defined. Furthermore, another big problem of Bitcoin and other crypto coins is the volatility of the price. For these reasons, the presence of scams is very frequent
- **Very complex:** Although this technology is revolutionary it is very difficult to understand because at the base there are very complex cryptographic mechanisms for the common man.
- **Blockchains can be slow and cumbersome:** Although being a disruptive technology, the Blockchain still has some inefficiencies especially on the timing of transaction processing. The normal payment systems, such as visa or mastercard, process more than one transaction per second per day. Bitcoin and Ethereum process respectively 6/7 and 15/20 transactions per second. This limit makes the Blockchain not yet adaptable as a standard payment system. In a context different from that of digital payments, as may be that of the IoT where the transactions are many, considering the number of connected devices, the problem of scalability would not guarantee the correct processing of transactions, as

there would be an amount in this network such transactions would create a bottleneck by making the network unable to process all these transactions.

2.3 2th-generation Blockchain technology

The second generation Blockchains are born with the concept of smart contract, a computer program that directly controls the transfer of digital currencies or assets between parties under certain conditions. The first Blockchain that allowed the execution of these smart contracts was Ethereum, which was developed by Vitalik Buterin at the age of 18. These smart contracts are written in a standard language, in such a way as to be understandable by the virtual machine, a program installed on every node of the network that allows the execution of smart contracts. Blockchain allows smart contracts to be deployed, written and executed within a decentralized, immutable, secure and reliable system. With this type of technology, contracts that were previously written on paper could be written in the form of a programming code including logics that would allow the execution of them. In the case of Ethereum they are written in a language called solidity and the virtual machine is called EVM (Ethereum Virtual Machine). This means that the developers can develop their own smart contract by going to insert logic at will according to their needs. These smart contracts have brought to birth a new type of application: the Decentralized Application (aka Dapp) or normal applications where either the front-end part or the back-end part interfaces with the Blockchain through these smart contracts (Fig 2.6).. Since in addition to generating immutable transactions, they are able to transfer value. For this reason, one of the most common use cases of these smart contracts is the implementation of tokens or programs that behave like coins. With the smart contracts, ICOs (initial coin offering) were born, that is a form of financing, used by startups or by subjects wishing to carry out a specific project, made possible through technology. In a nutshell, in order to obtain funding, it is proposed to the public (usually through a "whitepaper" cd) a project that will be realized through Blockchain with the creation of "tokens" to be sold, against a fee, to lenders.

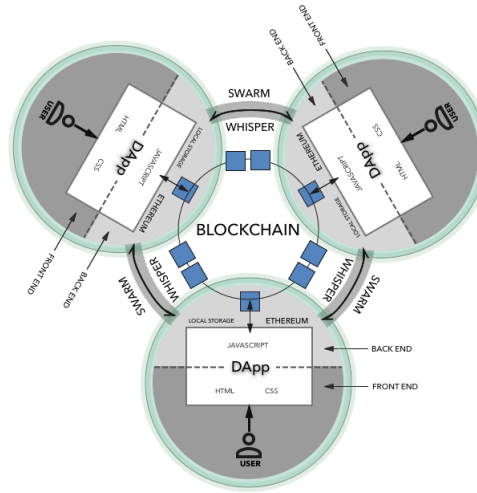


Figure 2.6. General Structure of Decentralized Application

2.4 3th-generation Blockchain technology

Unfortunately, the basic structure of the Blockchain brings great problems of scalability. For example, Ethereum is trying to solve this problem by implementing different solutions. One of the many is Plasma, which is based on the concept of sidechain, Blockchain that work parallel to the main Blockchain that extends functionality through interoperable Blockchain networks that allow a decentralized way to transfer / synchronize tokens between the two chains. Another possible solution to the problem of scalability is to revolutionize the basic structure of the DLT. For this reason the third generation Blockchains are born, ie dlt with a basic structure that is based on the concept of a DAG (Direct Acyclic Graph) rather than being based on a chain of blocks like the current one. One possible technology that uses graphs is IOTA. First of all, let's see what the acronym of DAG means and how it is applied to the concept of distributed ledger. The DAG (Fig. 2.8) literally Direct Acyclical Graph, is a data structure that has no direct cycles, that is, by choosing any vertex of the graph it is not possible to go back to it by going through the edges of the graph. A sequence can only pass from the first to the next and not vice versa. The DAG is often applied to problems related to data processing, scheduling, searching for the best route for navigation and data compression.

In short, therefore, it allows a different approach to create a distributed register. In the Blockchain, transactions are collected in blocks, which can only be created sequentially, thus forming a chain. In the DAGs, the "blocking concept as a set of transactions is not used",

exaggerated in terms of resources used. Since external miners are eliminated, transaction costs are reduced to practically zero. In this way, it is therefore possible to make micro-payments, and moreover, given the confirmation mechanism, as the number of nodes participating in the network increases, the number of transactions that can be confirmed and therefore performed increases, increasing the throughput of the network. whole network. A distributed ledger, however, must also guarantee the trustless nature of the system to be functional, and since we are talking about cryptocurrencies, it is necessary in particular that no double spending attempts are made and that transactions are executed quickly. The low transaction confirmation times are guaranteed by the blockless nature of the DAGs, since, as we said at the beginning, there is no need to create a block of transactions and propagate it on the network once verified. This process, in fact, in addition to limiting the number of transactions, has time constraints, given that, for example, a block is created every 10 minutes in the Bitcoin Blockchain. As a result, a transaction can never be confirmed in less than 10 minutes. In the DAG instead, since the transactions are propagated directly on the network, the confirmation takes place very quickly, just a few seconds. Obviously, this time depends on how much the network is populated, so initially a network entirely on DAG could turn out to be slower due to lack of nodes. A bit like what happened to IOTA in the first months of life of the network. As for double spending, which is the attempt to spend the same money over and over again, there are different approaches to avoid this happening. The concept of DAG itself requires that transactions be distributed according to a certain topological order, given the nature of the structure. Taking for example the implementation adopted in IOTA, suppose someone spends their money twice. In fact, when a double spending attempt occurs, two "branches" are created, but only one will then be correctly checked. To do so, IOTA uses the concept of "weight" during the transaction confirmation phase, which is why one of the two branches will grow faster than the other. In this way, the smaller branch, therefore with a lower weight, will remain isolated, thwarting the attempt to double the expense.

Obviously not even the DAG are perfect but different scientists and mathematicians are working a lot in the development of such systems in the field of distributed ledgers, given the different qualities compared to the classic Blockchain. One of the limitations of the DAG concerns the need to have a good number of active nodes in order to work correctly and quickly. Therefore, in networks that are not active or with few nodes, it may not work at its best

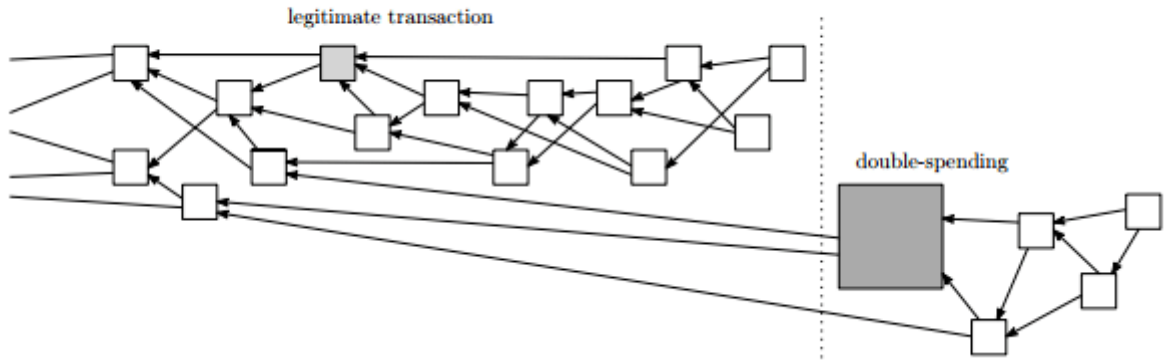


Figure 2.8. DAG double spending situation

and lead to confirmation times for transactions that are greater than the classic Blockchain. Precisely for this reason, also sudden changes in the number of nodes in the network could affect the functioning of the network. For example, a sudden drop in the number of active nodes would cause a decrease in the ability to confirm transactions, drastically reducing throughput. Furthermore, a correct consenting and "weighing" algorithm of the nodes must be implemented, in order to avoid illegitimate transactions and ensure correct scheduling of the transactions carried out. In short, the DAG is a very promising and already working solution, with further margins for improvement and development. There is no shortage of community uncertainties and doubts, but projects like IOTA show that this approach can work. It remains to be seen whether indeed such a system can practically climb to infinity as acclaimed by IOTA and other projects, but we can only answer this question by seeing how the network will behave with millions of nodes over the next few years.

2.5 IOTA

As is described in [2], IOTA is a German designed, next generation distributed ledger, which is based on a cryptocurrency called MIOTA (Millions of IOTA or Mega IOTA). The digital currency is specifically designed for the Internet of Things (IoT). IOTA aims to increase efficiency in the m2m economy which is based on the principle that the sensors or devices are able to send and receive funds without human presence. Sending and receiving IOTA tokens is guaranteed by the nodes present within the IOTA network. The name of these nodes is IRI. To send and

receive transactions, iota nodes create bundles To send and receive IOTA tokens, clients send packages of transactions called bundles to IRI nodes. The transactions in a bundle instruct the node to transfer IOTA tokens from one address to another. These addresses are generated by using a client's unique secret password called a seed. When the bundle is confirmed in the Tangle, the IOTA tokens are transferred. As previously mentioned, the basic architecture of IOTA is based on the concept of DAG and for this reason it brings great benefits thanks to this revolutionary technology

- **Trust:** All transactions in the ledger are transparent and immutable. Each IRI node in an IOTA network validates and stores transactions in its ledger, then sends its contents to other IRI nodes that do the same. As a result, all valid transactions are agreed on by all nodes, removing the need to trust a single one in the network.
- **Security:** IOTA uses quantum-resistant cryptography to secure the network and prevent attackers from stealing IOTA tokens. IOTA networks are peer-to-peer networks. No central authority controls the ledger of transactions, instead all IRI nodes hold a copy and run the software that contains the IOTA protocol to automate the agreement on its contents.
- **Cost saving:** Participation in the IOTA network is free, no fees need to be paid for sending or receiving transactions
- **Scalable:** For each transaction that's appended to the ledger, two previous transactions are validated. This process makes IOTA incredibly scalable because the more new transactions that propagate through the network, the faster other transactions are validated. Thanks to its potential, IOTA could be used in many industries, like for example: Mobility, Global trade and supply chains, Industrial IoT (Internet of things), Healthcare and Energy.

After many studies, the IOTA Foundation decided to adopt the ternary system instead of the binary one. This choice was made because it was discovered that this type of system leads to big savings from an energy point of view and turns out to be more efficient in resolving operations. In binary, data can be represented as either 1 or 0. These values are called bits.

Eight bits is equal to one byte, which can have 256

$$(2^8)$$

possible values. In trinary, data can be represented as 1, 0, or -1. These values are called trits. Three trits is equal to one tryte, which can have

$$27(3^3)$$

possible values. IOTA represents data as tryte-encoded characters, according to the tryte alphabet where each of the 27 characters consists of one tryte.

2.5.1 Tangle

The tangle [3] is a DAG-valued stochastic process where new vertices get attached to the graph at Poissonian times, and the attachment's locations are chosen by means of random walks on that graph. These new vertices (also thought of as "transactions") are issued by many players (which are the nodes of the network), independently. It is proved the existence of Nash equilibria for the system where a part of players tries to optimize their attachment strategies. So, The Tangle is the data structure that's formed by the connections among transactions in the distributed ledger on all IRI nodes. One of the validation criteria of a transaction is that each one must directly reference two previous transactions (tip transactions). This referencing model forms a type of directed acyclic graph (DAG), in which each transaction represents a vertex.

The data structure that forms the Tangle is a type of directed acyclic graph (DAG), and it was formally introduced in the IOTA whitepaper by Professor Serguei Popov in 2015. In the Tangle, transactions are connected to each other by reference in their `branchTransaction` and `trunkTransaction` fields. References form a hierarchy, whereby if one transaction is a child, the branch and trunk transactions are its parents. A reference can be direct or indirect. A direct reference is one that exists between a child and its parents. An indirect reference is one that exists between a child and any of its grandparents. The more direct or indirect references that a transaction has, the more likely it is to be chosen during tip selection. A transaction is considered confirmed when it's chosen during tip selection and one of its parents is a milestone, which was created by the Coordinator.

Coordinator

The Coordinator is a client application that sends milestone transactions to an IRI node at regular intervals. A transaction is considered confirmed when it's chosen by the tip selection algorithm and referenced by a milestone transaction. The Coordinator acts as a temporary safety mechanism until the majority transactions in a permissionless IOTA network are honest. An IOTA network relies on clients sending a majority of honest transactions to IRI nodes. However, the fewer transactions that are sent in an IOTA network, the easier it is for an attacker to send a majority of dishonest transactions to IRI nodes. As a result, an attacker may be able to double spend tokens, and carry out network-splitting attacks. To protect the network against these attacks, the Coordinator sends bundles of honest transactions to IRI nodes at regular intervals. These bundles include a signed zero-value transaction called a milestone. IRI nodes in an IOTA network reach a consensus on transactions that are directly or indirectly referenced by a milestone.

Tip selection

Each transaction in the Tangle must reference two previous transactions. Tip selection is the process in which an IRI node selects two random tip transactions from the ledger. These tip transactions are referenced by new transactions before being sent to an IRI node for validation. Tip selection is requested by clients so that they can reference two previous transactions in their new transactions. In general, the tip selection algorithm selects tip transactions that have no approvers. The tip selection process works as follows: the IRI selects a subgraph (also known as a subtangle) of the ledger and does two weighted random walks through it. Each weighted random walk returns a tip transaction hash. A subgraph is a section of the ledger that contains all transactions between a milestone transaction and tip transactions. The tip selection is done on a subgraph of the ledger to save computational power. The more transactions that the IRI node includes in the **weighted random walk**, the longer the tip selection process takes. For the tip selection process, the milestone transaction for the subgraph is defined by the client, and is calculated by doing the following:

$$latestMilestoneIndex - depth \tag{2.2}$$

The result of this calculation is equal to the index of the milestone transaction that is used to form the subgraph. The higher the value of the depth parameter, the more computations the IRI node must do. To restrict the value of the depth parameter, IRI nodes can change the MAX-DEPTH configuration option. A weighted random walk is an algorithm that the IRI uses to find a path to a tip transaction in a subgraph. To increase the probability of selecting a path to new transactions, the algorithm favors a path through transactions that have a higher rating. This rating is called a cumulative weight. The cumulative weight of a transaction is calculated using the following variables:

- **Future set:** Transactions that directly or indirectly reference the transaction
- **ALPHA configuration parameter:** A number that affects the randomness of the tip selection process

The IRI gives a high rating to a transaction with a large future set because it has a higher probability of being confirmed than one with a small future set. However, if the IRI were to rate transactions based only on this variable, the ledger would become a long, narrow chain of transactions, which are referenced by many other transactions. This would slow the rate of new transactions being appended to the ledger because new transactions would have to wait until they had a large enough future set before other transactions would reference them. So, to increase the speed at which new transactions are appended to the ledger, the IRI also uses the ALPHA configuration parameter to calculate the cumulative weight. The ALPHA configuration parameter makes sure that the cumulative weight of each transaction is calculated with an element of randomness. This parameter allows the IRI to select some transactions that have a small future set and by doing so, increase the speed at which new transactions are appended to the ledger. The tip selection algorithm is not enforced. Instead, IRI nodes have an incentive to use it to have the best chance of their transactions becoming confirmed. It's impossible to check if an IRI node used the tip selection algorithm or even changed it to return custom tip transactions for its own purposes. However, it's necessary that tip transactions are selected at random. This randomness is important because if IRI nodes chose to select the best tip transactions (for example, those that approve their own transactions), only a few lucky transactions will ever be confirmed.

In Figure 2.11, the black transactions are considered the best, and each new transaction tries to reference them. This situation reduces the rate at which all transactions are confirmed. When all IRI nodes use the tip selection algorithm, all tip transactions are selected at random. This randomness increases the rate at which new transactions are attached to the Tangle and eventually confirmed. When all IRI nodes use the tip selection algorithm, all tip transactions are selected at random. This randomness increases the rate at which new transactions are attached to the Tangle and eventually confirmed (Fig. 2.12).

Proof of Work

Proof of work (PoW) is the answer to a mathematical problem that's difficult to calculate, but easy to verify. In IOTA, proof of work protects the network from spam transactions. PoW is calculated using trial and error, therefore it requires the use of computational power. Originally, PoW was introduced as a concept to reduce large amounts of email spam. This concept is known as hashcash, and is a method of preventing email spam by requiring a proof of work for the contents of every email. Similar to hashcash, each IOTA transaction must include a PoW before it can be validated. This PoW provides spam protection for an IOTA network by increasing the time and computational power it takes to create a valid transaction. If an IRI node receives a transaction without the PoW, that transaction is ignored to reduce the effect that spam transactions have on the network. PoW can be done by clients or it can be outsourced to an IRI node. Clients may want to use remote PoW if the device they're using to create transactions doesn't have the necessary computational power to calculate PoW in a reasonable amount of time. To calculate the PoW for a transaction, the following contents of the transaction are converted from trytes to trits, then those trits are hashed:

- **Bundle hash:** Hash that is calculated using the address, obsolete tag, timestamp, value, and index of all transactions in the bundle
- **Signature:** Signature of the transaction (if it debits IOTA tokens)
- **Trunk transaction and branch transaction:** Two previous transactions that the transaction references and approves

If the hash ends in the correct amount of 0s (minimum weight magnitude), it's considered valid. If the hash doesn't end in the correct amount of 0s, the value of the transaction's nonce field

is incremented and the hash is hashed again. This process continues until a hash is found that ends in the correct amount of 0s. The nonce field of a transaction contains a string of 27 trytes that IRI nodes use to validate the PoW. The minimum weight magnitude (MWM) is a variable that defines the number of 0s that a transaction hash must end in. Although the MWM is set in the node software (IRI) of each IRI node in an IOTA network, clients and IRI nodes can choose to change the MWM. However, if each IRI node used a different MWM, transactions would be valid only for those with the same MWM. This situation would decrease the rate of transaction confirmations. Every increment of the MWM increases the difficulty of the PoW by 3 times.

2.5.2 Basic Concepts

Each client in the network must create a seed, which is used to create private keys and addresses. Addresses are public and are what clients send data and/or IOTA tokens to. The private key is secret and is used to prove ownership of an address. Clients send data and/or IOTA tokens to each other's addresses in packages of transactions called bundles. To prove ownership of an address, clients use its private key to sign bundles that debit IOTA tokens.

Addresses and signature

Seeds are used to create private keys, and in turn, private keys are used to create addresses and signatures. Addresses are public and can safely be shared among others in the network. Signatures are cryptographic proof of ownership of an address. So, when a transaction contains a valid signature, it was created by the owner of the correct seed. Each private key is created from a cryptographic hashing function that takes a seed, an index, and a security level. Private keys are used to create addresses, so the index and security level of the private key also affect the address. A private key can have one of three security levels, which determine its length. The higher the security level, the longer and more secure the private key. As well as a security level, a private key has an index. Whenever the index is incremented, a new private key is created. Private keys, like seeds, are secret and should never be shared with anyone. The private key is created as follows: The seed and index are combined and hashed to create an 81-tryte subseed which is passed to a cryptographic sponge function that absorbs it and squeezes it 27 times per security level. The result of the sponge function is a private key that consists of 2,187, 4,374, or

6,561 trytes, depending on the security level. Regarding the addresses, they are public. Clients send tokens and messages to them using the address field of a transaction. A private key is used to create an address with the same index and security level. As a result, the same seed, index, and security level will always result in the same private key and address. To create an address, the private key is split into 81-tryte segments. Then, each segment is hashed 26 times. A group of 27 hashed segments is called a key fragment. Because a private key consists of 2,187, 4,374, or 6,561 trytes, a private key has one key fragments for each security level. For example, a private key with security level 1 consists of 2187 trytes, which is 27 segments, which results in one key fragment. Each key fragment is hashed once to create one key digest for each security level. For example, one key fragment results in one key digest. Then, the key digests are combined and hashed once to create an 81-tryte address (Fig. 2.13).

Clients need a way of proving that they own an address before IRI nodes will validate a transaction that debits IOTA tokens from it. To prove ownership, input transactions must be signed with the private key that was used to create the address. A signature is created from both the private key of an address and the bundle hash of the transaction that spends from the address. By using the bundle hash to create a signature, it's impossible for attackers to intercept a bundle and change any transaction without changing the bundle hash and invalidating the signature. Signatures are created using the Winternitz one-time signature scheme. This signature scheme is quantum resistant, meaning that signatures are resistant to attacks from quantum computers. To create a signature, the bundle hash of a transaction is normalized to make sure that only half of the private key is revealed in the signature. With this kind of signature it is important to adopt the **address reuse** technique: This step is necessary because of the Winternitz one-time signature scheme. If the bundle hash weren't normalized, the scheme would reveal an unknown amount of the private key. By revealing half of the private key, an address can safely be debited from once. If an address is debited from more than once, more of the private key is revealed, so a sophisticated attacker can brute force its signature and steal the IOTA tokens. Clients need a way of proving that they own an address before IRI nodes will validate a transaction that debits IOTA tokens from it. To prove ownership, input transactions must be signed with the private key that was used to create the address. A signature is created from both the private key of an address and the bundle hash of the transaction that spends from the address. By using the bundle hash to create a signature, it's impossible for

attackers to intercept a bundle and change any transaction without changing the bundle hash and invalidating the signature.

Bundle and Transaction

A transaction is a single instruction to credit IOTA tokens, debit IOTA tokens, or send a message. To transfer IOTA tokens, you need input and outputs transactions, which are grouped together in a bundle. Each transaction that's sent to an IRI node, even individual zero-value ones, must be packaged in a bundle. You may need more than one transaction for the following reasons:

- To transfer IOTA tokens from one address to another, you need both an input transaction to debit IOTA tokens from the sender, and an output transaction to credit IOTA tokens to the recipient. These transactions must be indexed (starting from 0) and packaged together in a bundle.
- An input transaction, which debits IOTA tokens, must contain a signature in the `signatureMessageFragment` field. If the security level of the address is greater than 1, the signature is too large to fit in one transaction and must be fragmented across zero-value output transactions.
- The `signatureMessageFragment` field of a transaction can contain a limited number of trytes. Therefore, to send a message that exceeds the limit, you must put the rest of the message in another transaction (transaction index 1).

A bundle can consist of any number of input and output transactions. We recommend a maximum of 30 transactions in a bundle. Input transactions debit IOTA tokens from addresses. Bundles can contain multiple input transactions, and each one must include a signature. The length of the signature depends on the security level of the address. Output transactions can be a zero-value transactions that contains only a message or transaction with a positive value that credits IOTA tokens to an address.

2.5.3 MAM: Masked Authentication Message

IOTA is the first distributed ledger that has no transaction fees and is built for the Internet of Things ecosystem; promoting a future in which the devices are able to make and receive payments on their own. To propagate transactions on the network, IOTA uses the Gossip protocol. This mechanism means that any data with sufficient weight can be dispersed to the opposite side of the cluster efficiently. The concept of a MAM channel is very similar to the concept of a channel on Youtube where there are many viewers and a publisher. The latter is the one that publishes the data while the viewers after registered entities will be able to receive the data issued by the publisher. Thanks to MAM a user can publish data when he wants. The only thing that needs to be done is to perform a small proof of work in order to correctly propagate the transaction. If nodes are listening for the channel ID in real time, the message will be received by a subscriber when it reach his node. These messages can have any size but studies have said that the more the message is small the better it is propagated. For example, one user can transmit an encrypted HD video using MAM but this will saturate the network leading to a lagging user experience. Therefore it is more efficient to use MAM to signal some off-tangle protocol for the actual streaming. Since these messages are part of the Tangle, they both contribute to the security of the network by increasing total hashing power and benefit from the data integrity properties of the network as other transactions continue to indirectly reference them. MAM uses the Merkle tree based signature to sign the digest of the encrypted message. The root of the Merkle tree is used as the channel identifier. Because each tree lasts for a short time, each message contains the id of the next root. The messages are encrypted with a one-time pad consisting of the channel ID and the index of the key used to encrypt. The result of this process is encrypted using the private key of one of the leaves. The encrypted message is published on the tangle where anyone knowing the symmetric key can find and decrypt it.

When a stream is consumed on the MAM channel, the message is first authenticating by checking the signature and verifying that it belongs to a leaf of the tree and finally unmasked. if the verification of the signature fails, the whole message is considered as invalid. MAM channels can be used in different ways:

Public

In public mode, the root of the tree is published as the address on which it is possible to receive the published message. A user who is in possession of the root is able to decrypt the data. This process resembles a broadcasting process on a radio channel. It can be used for public announcements directly from a device or an individual. A possible use case could be a twitter simulation with the properties of data immutability and integrity (Fig 2.16).

Private

Private mode can be used for encrypted data streams that should not be made public. In this mode, the root hash is used as an address, this allows random users to not be able to decrypt the message because it is not possible to derive the root from the hash due to the non-invertibility of this function. This means that the data issued on this channel can only be issued by those who have permission. This mode is similar to a system of encrypted radio in which everyone can see but nobody is able to understand. Furthermore, this mode is useful for private communication between devices.

Restricted

Restricted channels instead add a key to the private channel. The address used is the hash relative to the root of the merkle tree and the private key. This way a publisher could stop using the authentication key without affecting the channel id. Basically, access could be revoked by the subscriber if desired. When a key change occurs, the new key must be distributed to users who can follow the stream.

Since each message contains the root of the next merkle tree, it is easy for a user to follow the stream after having received a valid address for the first time, and if the channel is also restricted, the key. Since a message points to the next merkle tree, there is no way for a user to receive previous messages to the root address that was provided to him. In addition, the MAM publisher can split the channel whenever it wants. This means that future messages may belong to a new tree whose root will not be exposed before. This allows you to create channels related to data types that do not want to be shared. A further approach would be the subdivision of data by short-term channels. When a device notices the presence of an anomaly, it can split

the channel and start publishing data at smaller intervals. In addition to continuing to publish data on the main channel, it can signal to subscribers the relevant channel split. This allows the flexible creation of secondary channels which can be used for different use cases. For this we can say that Masked Authentication Message is one of the most powerful modules built on IOTA that opens the door to many use cases. Being able to secure data's integrity and control its access management is a prerequisite for things like Over-The-Air updates, Fog Analytics, Data Marketplaces, Automated Insurance, End-to-End verifiable Supply Chains and so much more.

2.5.4 Qubic: Quorum-based Computation

Qubic is a protocol that uses IOTA's solution for quorum-based computations, including such **oracle machines, outsourced computations** and **smart contracts**. A quorum is the minimum number of votes that a transaction must obtain in order to be approved and to perform an operation on the distributed network.

A quorum-based technique is implemented to enforce consistent operation in a distributed system. Qubic allows people to take advantage of unused computing capacity while helping to protect the IOTA Tangle: Individual qubics are pre-compiled and pre-packaged computational processes. Qubic leverages the tangle to pack and distribute qubics from their owners to the oracles that will process them. Qubics are published as messages in normal transactions and contain special instructions, called metadata, on how and when to process them. the oracles can read the qubic metadata inside the transactions to decide whether or not to perform the requested task The IOTA nodes enabled for Qubic (in short the Q nodes) can handle most of this decision automatically based on the parameters set by their users. These operations are written using a functional programming language called Abra, which is based on ternary system. The model on which Qubic is based allows the flow of these qubic within the tangle. From a technical point of view, qubics are performed thanks to events, which will generate input on which the execution of these qubics will be based. Since qubics post their results on Tangle, performing a single qubic can cause many other qubics to perform in a sort of "chain reaction". Basically, qubics can live on the Tangle in a dormant state. As soon as a specific data is changed, the qubic will generate an event which could "awaken" other qubic. This allows a highly dynamic programming environment because you can add new qubics at any time and link them to any

input data you like. In addition, anyone can publish a transaction containing the qubic code he wants to perform and, for appropriate rewards, see the results displayed in the tangle.

Oracle Machine

In a perfect world there would be only honest oracles that could be delegated by ourselves to perform calculations, but unfortunately it is very likely that there are evil actors who try to take advantage of others by influencing the results. This is why we need to add protection against evil actors. The way this is implemented in the Qubic protocol is by forcing quorum consent. Since it is impossible to expect that each oracle performs all the existing functions on the tangle and verifies all the results, qubic uses the quorum to reach consensus on the results. This means that in the Qubic system a group of oracles will join an assembly in which all members will process the same data along with the other qubics and each oracle will publish the processing results for each qubic on the tangle. This allows qubic owners to determine consent to the quorum of the assembly. If it is not possible to form a quorum (at least $2/3$ of the oracles in the assembly agree on the result), the results will not be accepted by the owner of the qubic. For this reason it is necessary to develop an incentive system to be honest in an assembly. If the owner of the qubic agrees with the data generated by the assembly, he will offer a reward. This reward will be distributed to the oracles in the assembly that produced the quorum result for that qubic. In case the result was wrong, every oracle will lose the reward .. The same is true if you do not decide not to participate in the processing of the qubic. In case there were no elaboration there would be no prize. Note that a qubic owner can increase confidence in the processing results by selecting a larger assembly, or perhaps even the qubic processed by multiple random assemblies. It all depends on the importance of the data and the amount of reward it wants to spend on spending.

Qubic lifecycle

The first step is to prepare a qubic, the owner generates the Abra code and the qubic metadata within an IOTA transaction which is called **qubic transaction**. In this regard, Qubic uses the mechanism generates a transaction with a value of 0 by inserting this data into it. At this point the owner chooses which assembly will treat the generated qubic by looking for transactions that provide information about oracle assemblies. These are known as assembly transactions

which inform the oracles that the qubic is available for processing. The oracles, then, are the Q-node operators who will process the qubics. Qubic transactions must be well-formed and signed, otherwise the nodes will refuse to process them. Once a qubic transaction has been validated, the node will prepare the qubic for execution on the particular node hardware and schedule its processing. Once the qubic has been processed, the quorum is reached and the results are written on the Tangle. In this situation two facts can happen: the qubic returns pending and waits for changes related to the inputs that give the trigger, or generate a cascade effect so that other qubics operate on the new inputs provided.

ABRA

Abra is a functional programming language based on trinary system, developed to be compatible with the Qubic protocol which, like the IoT itself, is designed to work on a wide variety of hardware. To enable the execution of the same code on different hardware platforms, qubics is contained in this language, which means that this language lends itself to easy translation or interpretation for a specific hardware. This allows Qubic to be independent of the hardware it will operate on. A functional programming language allows a simpler analysis to prove the correctness of the code. In addition, functional programs lend themselves to intensive use of parallelization, which means that different pieces of a larger program can run simultaneously to exploit more CPUs or even more devices. Finally, Abra is based on ternary systems which can provide significant energy savings, a very important consideration in the context of the IoT. A ternary digit, a trit, can represent 1.58 bits. The amount of energy needed for a ternary system can be reduced to about 65% of an equivalent binary system, resulting in corresponding energy reduction.

Qubic Protocol

An assembly is shared by publishing a series of parameters on the Tangle in the form of an IOTA transaction. This operation can then be used by an oracle to find the assembly and decide whether to join. Assembly sets composed of more than two untrustworthy oracles will most likely be the norm in particular for qubics to be processed publicly by strangers. This model gives priority to decentralization and trust at the expense of latency. The new oracles will not be able to join the assembly at random times. However, an assembly can give the possibility

to join the new oracles at certain predefined time intervals, called `textbf` epochs. During an era, the oracles that make up the assembly are all known. Within each era, all oracles will process the same set of data and then activate the same set of qubics. Each era has an exact time and duration, which are published as global parameters through a transaction on the Tangle. The epoch transaction also describes if and how other oracles can join the assembly during the era. The Qubic process is essentially divided into 2 phases:

- **Resource test:** To prevent the possibility of a Sybil attack, assemblies require oracles to provide evidence of their resources. The Qubic system is very flexible in this regard and can accept many ways to demonstrate resources. It also allows a mix of different types of tests, depending on the assembly needs. The oracles that want to join the assembly will have to provide specific evidence of resources during a so-called initial phase of testing for each age. The parameters will specify how long the resource test phase will last and what kind of evidence must be provided. The results will show that each oracle is a separate entity and also the relative capacities of each oracle in the form of a weight factor.
- **Processing:** Once the test phase is complete, the oracles participating in the assembly will begin processing qubics that have been linked to the assembly. Because of the functional nature of the Abra intermediate language, a qubic will always generate the exact same output result data as the same input data. Therefore a qubic will process its inputs and eventually end with a stable output state. The time taken by a qubic to completely process its inputs is called **quant**. Note that a quant is measured in logical time, not in real time, because the actual processing time for each qubic may vary. Abra does not allow infinite loops and for this reason qubics must absolutely always end in a finite time. The inputs of a qubic have a time limit related to the default call within a quant in order to avoid infinite loops. A quant term when all qubics have processed their inputs and generated their results. Because of these constraints imposed by Abra, it is important to keep in mind that a qubic is not Turing-complete within a single age. However, Qubic allows the completeness of Turing over an indeterminate number of epochs.

2.6 IoT and Blockchain: applications and problems

The Internet of Things (IoT) mixes virtual and physical worlds in the creation of intelligent environments. With the progressive expansion of the business linked to this type of environment, however, the technological challenges and the implications linked to the security and interoperability of increasingly IoT architectures are multiplying. This is why many people think that distributed trust technology is the only one able to ensure scalability, respect for privacy and reliability for growing IoT environments. According to report by Cisco, in 2020 there will be 50 billion devices online. With so many connected objects, all of which need to send and receive instructions to turn on, shut down and work, the amount of data transiting (and sometimes clogging) corporate and public networks is bound to explode, with management costs that do not have had equal in the past. Other problems are how exactly you will be able to monitor and manage billions of connected devices, store the metadata that these devices produce and do all of this in a reliable and secure way. The Blockchain is a candidate for the key role of the application for the industrial IoT. In fact, this technology can be used to track billions of data emitted from connected devices, allow processing of them and coordination between physical devices. This decentralized approach would eliminate the breaking points of traditional networks, facilitating the creation of a more stable, secure and transparent ecosystem on which smart devices can operate. Furthermore, the cryptographic algorithms used by the Blockchain would increase the data protection of private consumers. The Internet of Things (IoT) represents the first step towards the full digitalization of our society, especially if combined with this technology to create a network in which all the billions of objects we all use will be interconnected with each other through sophisticated communication networks, safe and efficient. The decentralized and secure nature of Blockchain make it an ideal technology for communication between the individual nodes of an IoT network, so much so that it has already been embraced by some of the leading brands in the field of corporate IoT technologies. The coupled Blockchain-IoT seems to bring so many benefits. A possible example is that of the supply chain (system of organizations, people, activities, information and resources involved in the process of transferring or supplying a product or service from the supplier to the customer), in which the presence of sensors that constantly produce immutable data on the state of a product turns out to be a better effective of the current process. This combination is also useful in the context of smart

cities. The binomial Blockchain and IoT still presents some problems today. The first problem is that of scalability, if we think that in 2020 there will be about 50 billion of connected devices, which every second will produce data that will be written on the Blockchain, at present, the Blockchain is not able to process such a large number of transactions. Another huge problem related to this combination is the problem of data saving. Because the Blockchain is based on the fact that every node in the network has to keep a copy of the history, in a context like the IoT, devices would not be able to store such a large amount of data at this time.

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

Figure 2.9. trinary numeric system

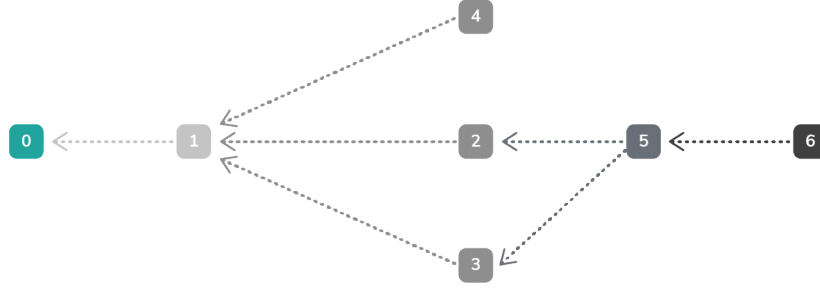


Figure 2.10. DAG

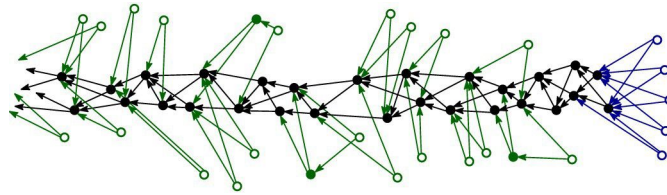


Figure 2.11.

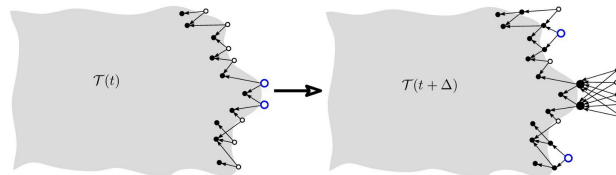


Figure 2.12.

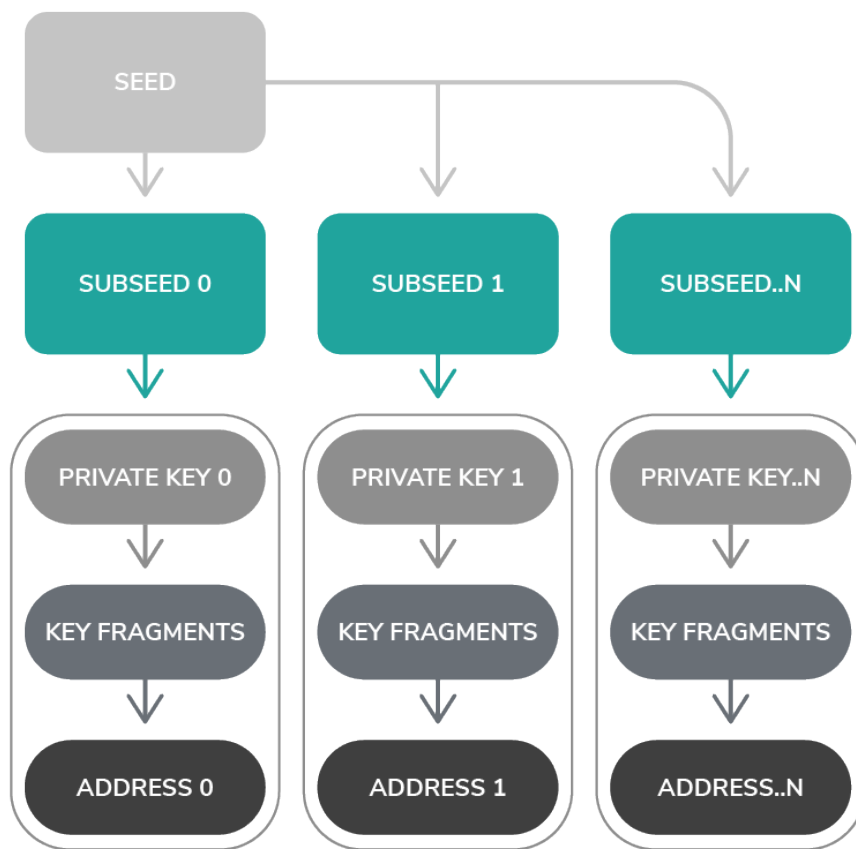


Figure 2.13. Address Generation

Transaction index	Transaction contents	Transaction value
0	Recipient's address	80 (sent to the recipient's address)
1	Sender's address, and the first part of the signature	-100 (the total balance of the sender's address as a negative value)
2	Sender's address, and the second half of the signature.	0
3	An address for the transfer of the remaining IOTA tokens (usually one of the sender's addresses).	20 (Remainder of the sender's address in transaction 1)

Figure 2.14. bundle of a transfers 80 IOTA tokens to a recipient from an address with a security level of 2.

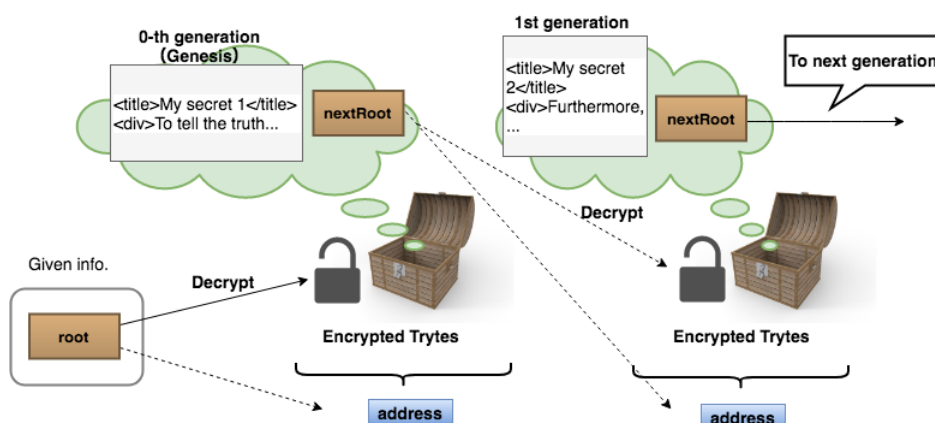


Figure 2.15. MAM Public mode

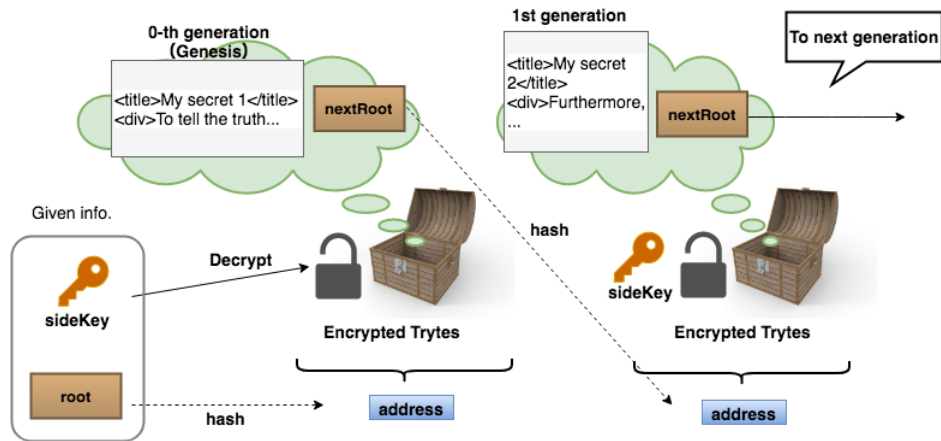


Figure 2.16. MAM Restricted mode

Chapter 3

Development

The present thesis work is based on the development of an extension for google chrome that implements a decentralized marketplace of data emitted by sensors connected to the IOTA network. The end user is able, thanks to the use of this extension, to send and receive transactions, monitor their balance, and most importantly, to buy streams of data emitted by the devices. Since these data streams need to be written on the iota Tangle in order to be available to the final user, a program has also been developed that can take data from the sensors and write them on the IOTA Tangle thanks to the use of the MAM protocol described previously. Therefore, the development of this thesis work is made up of two applications: the extension for Google Chrome installed on end users' PCs and a program able to collect data from the sensors and write them on the Tangle. Google Chrome extensions are composed of different but cohesive components that include content scripts, background scripts, an options page, user interface elements, and various logical files. Extension components are created with Web development technologies: HTML, CSS and JavaScript. These components of an extension will depend on its functionality and may not require all options Regarding the first, that is the application that allows you to interact with the marketplace, has been developed using the most used development coupled in the cryptospace, React and NodeJs while for the demon, it was enough only the use of NodeJS as this application does not have a graphical interface. The creation of Web applications, regardless of the framework chosen for development, necessarily involves the three fundamental languages of the platform: HTML for the structure, CSS for the stylization and JavaScript for the application logic. React is a Javascript library created by Facebook and Instagram that allows you to create a single-page application (Single Page Applications (SPA))

through a structure that divides it into dynamic and reusable components. A React component is a JS class that extends the Component class provided by React. A component represents and defines an atomic block of HTML code, along with any dynamic behaviors associated with that fragment (for example, click events). They are like small Lego blocks, which can be composed at will to each other up to realize complex applications at will. A component is isolable and reusable, and is composed solely of Javascript methods. The fundamental method is to render () which simply returns the piece of HTML to show. Although declarative, the representation of the component actually translates into calls to the React API which intervene - in the fastest and most efficient way possible - on the DOM of the page to create the necessary elements. The syntax used to define the components is called JSX: it was developed by the creators of React to facilitate the interaction between Javascript code and HTML markup within the components. The code written in this syntax needs to be then compiled, so as to become entirely Javascript code. Instead, Node.js is an open-source platform that allows you to run applications written in Javascript outside the browser. For this reason, Node.js incorporates within it the Javascript V8 engine, built by Google and also used within the Chrome browser. V8 allows to convert and execute the code written in Javascript language. Before the advent of Node.js, Javascript was mainly used within the browser to write scripts to be executed within web pages. Node.js is the **libuv** multiplatform library (written in C language) that is used for the asynchronous management of I / O operations. To do this, **libuv** implements a loop that takes the name of the Event Loop. This loop is used to manage the different I / O operations (read / write files, database access, network requests) in an asynchronous and non-blocking manner, guaranteeing excellent performance and providing developers with an intuitive API and simple to use. A possible alternative to this solution would be to run the code synchronously. In this case, however, every time a resource is requested, the rest of the program waits until the resource is available and it is not possible to execute other instructions. Another method, certainly more effective than the one just described, is to make use of the concurrent programming technique through the use of multiple threads. Although the latter solution is more efficient than the previous one, using several threads that access shared resources can be more complicated. This explains why Node.js uses the Event Loop. Because thanks to the latter Node.js ensures the execution of I / O operations in an asynchronous and non-blocking manner, ensuring on the one hand a high efficiency and allowing the other to take advantage of the simplicity of a language such as

Javascript that, through the V8 engine, executes only one instruction at a time. Every time we start the execution of a script, Node.js automatically initializes the Event Loop which will then be terminated when the script is completed. The Event Loop is used by Node.js to implement the asynchronous and non-blocking model of I / O operations management. In fact, remember that every time a certain resource is requested, Node.js, instead of blocking its execution and waiting, delegates the underlying operating system that can handle several operations simultaneously. When the operating system has completed a certain operation, it informs Node.js by generating an event. These events will be placed inside a queue and the respective callback functions (recorded during the writing of the program) will be executed one at a time when the V8 engine call stack is empty, ie when V8 is not executing other instructions. When writing our programs, we will indicate through callback functions which code to execute whenever a certain resource is ready to be used. In this way it will seem that several instructions are executed simultaneously when in reality V8 is able to process only one at a time. Thanks also to Node.js, today Javascript is one of the most widespread programming languages used in various fields. Along with Node.js was also introduced NPM, a package manager for Javascript that allows developers to organize, reuse, publish and share the source code. NPM is the main software used to handle Node.js modules and lets you share code for typical problems among JavaScript developers. The philosophy behind NPM is that if a problem has already been solved by other programmers it makes no sense to develop it on its own, you can use the shared solution made available to everyone. NPM besides allowing the reuse of the code, allows you to keep it constantly under control in order to update it if it should be improved. NPM divides the code into packages or modules. A package is a directory that contains one or more files together with a file called `package.json` that contains data about the package. Typically, a typical application consists of tens or thousands of packages. For the implementation of the HTML layout, the *Bootstrap* version 4.0 library was used. Bootstrap is a set of graphic elements, stylistic, layout and Javascript ready to use, born within Twitter by developers Mark Otto and Jacob Thornton. Today Bootstrap is an independent project and has been made available to developers around the world who are free to use this framework as a basis for their web projects. To date, Bootstrap can boast a truly global success: strong of an audience of rapidly expanding users, to date Bootstrap is one of the most used solutions for web designing, above all in a responsive approach. This framework offers us the building blocks to build HTML5 web pages, completely

responsive, coherent and functional. The Bootstrap utility is immediately evident, especially in the current situation where web pages can be enjoyed on a myriad of devices with different characteristics. Bootstrap will take care to provide you with elements of style that allow the page to adapt to the device using, at the same time, interface elements common to modern sites, those that the user expects and of which he knows behavior and meaning. It is possible to divide bootstrap into 4 main elements:

- **Scaffolding:** This area contains all those CSS elements that allow us to define the page structure, ie its layout. The constitutive part of this page is the Grid system, that is a fixed or fluid grid, with a base width of 960px in which rows and columns can be defined in which the contents are then enclosed. Basically, the grid consists of 12 columns and each element of the layout can expand on one or more columns, simply by applying a style class that somehow defines its "extension".
- **CSS:** This area contains predefined styles for different elements of the page, such as titles (H1, H2, ...), tables, buttons, form elements, images. With these style rules make buttons of various sizes, with beveled edges, with an over effect becomes really simple and immediate. The same applies to the creation of a table with alternating color lines.
- **Components:** This area contains more complex elements of buttons or tables, but now very common in websites. For example, groups of buttons, navigation bars, drop-down menus. The components also include a set of commonly used icons made available by Glyphicons. The Glyphicons icons are usually paid, while the test distributed with Bootstrap is free. The creators of Bootstrap suggest to those who use them to insert, if possible, a link to the Glyphicon site, as a kind of thanksgiving.
- **Javascript:** This last area contains several jQuery plug-ins to achieve very common effects such as transitions, modal windows, popups, carousel, accordion, tab. Even these plug-ins are easy to use and allow you to create many interesting solutions.

It is the most popular project on the GitHub platform, with over 58,000 stars and more than 20,000 forks. Since this application has been made open source, the entire application code has been shared through the *GitHub* platform. GitHub is the heart of a Git repository hosting service, which is cloud-based source code management. GitHub also implements functionality for

code review (pull requests, diff and requests for revision), project management (including monitoring and problem assignment), integrations with other development tools, team management, documentation and "social coding" ". Much like a sort of social network for developers, GitHub is an open environment where programmers can freely share and collaborate (even ad hoc) on open source code. GitHub makes it easy to find useful code, copy repositories for personal use and send changes to other projects, becoming the site of virtually every open source project of any importance.

3.1 Architecture

This system is called Pegasus, and, as previously written, is composed of two applications called PEGASUS-APP and PEGASUS-DAEMON. This system is called Pegasus because I imagined a smart city as a collection of objects connected to each other almost as if it were a constellation. That's why I decided to call it Pegasus. As for the first application, the one that allows you to interact with the sensors installed within the city, has different features. The first one implements a decentralized marketplace in which it is possible to buy data directly from the sensors installed in the city. Moreover it implements all the functionalities of a wallet, because these features are intrinsic to the fact that in order to buy data streams it is necessary to have funds, which are managed directly by the application itself. So as described in the previous chapter, a cryptocurrency wallet must allow the user to send and receive tokens and monitor its balance. The most delicate functionality in implementing a wallet is key management. In the case of IOTA, the most delicate part is the management of the seed because in order to perform operations it is necessary to supply the seed. The second application, is the one that must be installed on sensors belonging to this marketplace, ie those sensors that allow sending data to users who have purchased this right. Each sensor connected to the IOTA network sends the data detected on the IOTA Tangle thanks to the use of the MAM protocol. The sensor, thanks to this protocol, creates a restricted channel which will provide the key to decrypt the content of the data, to the users who will make the payment to the sensor to have visibility of the data. Since this type of channel is based on symmetric cryptography, ie the data is encrypted with a symmetric key, a user must possess the key in order to receive correct data. So to be able to send this key, is used the asymmetric encryption: when the user pays the sensor, in addition to

sending the amount for which you can access the data, he also sends his public key so that the sensor after receiving the amount necessary to allow him to receive the data, will send him the key of the MAM channel encrypted with the public key just received, so that only the user who actually paid will be able to decrypt the access key to the MAM channel with his private key.

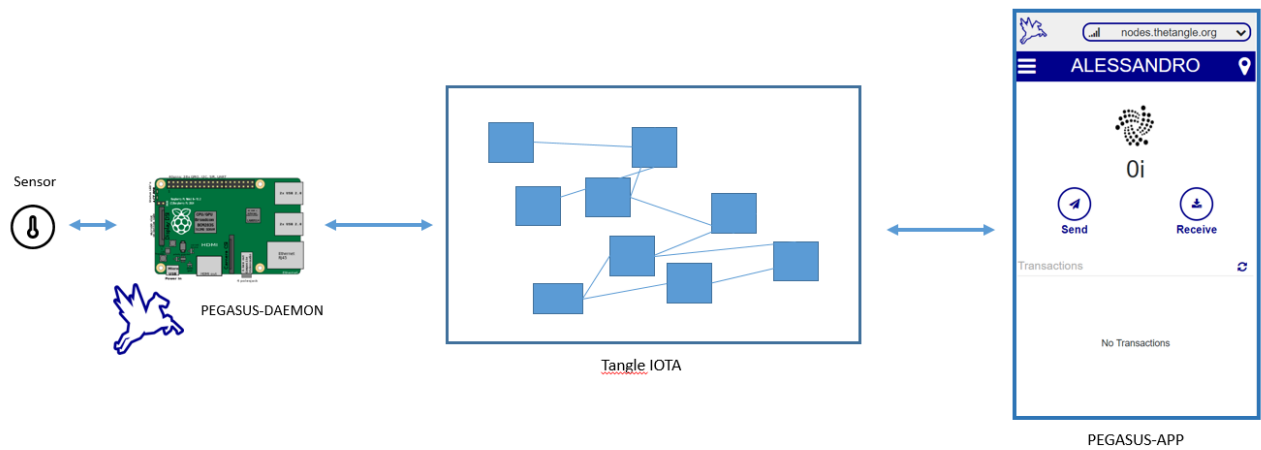


Figure 3.1. PEGASUS Architecture

3.2 Functionalities

In this thesis work, in order to better interact with the IOTA network, IOTA's official javascript libraries was used. These libraries allow you to create transactions, sign transactions, generate addresses and inter-connect with the other nodes connected to the IOTA Tangle. As it would be difficult and economically difficult to test the correct functioning of the system, IOTA Foundation offers two types of networks. The first one is the Mainnet, which is the original blockchain in which the actual transactions take place in the distributed ledger and the native cryptocurrencies have a real economic value. In other words, the Mainnet refers to the true open source Blockchain that is publicly consultable and verifiable. Having a Mainnet means letting you know that the project is now live and is in technical in progress. In addition, a live Mainnet would test the functionality and capabilities of the Blockchain, since the public

is able to participate freely in the network and any malfunction could compromise the entire operation of Blockchain. Therefore, starting a Mainnet requires a significant amount of technical resources to ensure that each component functions as it should without the presence of bugs, which in this case could cause major damage. Beyond that, a Mainnet serves as proof of work that the project is performing its vision well. Instead, Testnet is an exact replica of the original Blockchain, with the same technology, software and functionality. The only difference is that the transactions on the Testnet are simulated and the coins within of this have no real economic value outside the Testnet environment. The Testnet is a simulated environment in which the functionality and capabilities of the Blockchain are constantly optimized and tested by application developers and testers so that they can then be made on the Mainnet. Transactions on the main network are "false" because they are trial transactions, with no transaction costs incurred and no implementation costs required by the developers. Since the coins on the Testnet are useless, there is no economic incentive for mine miners because their sole purpose is to facilitate the transaction testing.

Initialization

This process consists of 3 main steps: the first (Fig 3.2) is the creation of the account name. The application will allow you to continue to the next step only if the user actually inserts a name in the appropriate text input. Regarding the second registration step (Fig. 3.3) the user must enter his password which must be at least 8 characters long. As long as the user does not enter the password he chooses twice in the appropriate spaces, he will not continue with the registration. In order to continue the entered passwords must be the same. Once the password is entered, the password hash is made and the result of the hash function is saved to the local storage. The choice to save the password hash and not the password itself was dictated by avoiding saving it in a comprehensible version in order to avoid, in case of a hacker attack, that the password is understandable. The hash function used for this purpose is SHA-256. The term SHA (acronym of the English Secure Hash Algorithm) indicates a family of five different cryptographic functions of hash developed since 1993 by the National Security Agency. Like any hashing algorithm, the SHA produces a message digest, or "footprint of the message", of fixed length starting from a message of variable length (in this case of 256 bits). The security of a hash algorithm lies in the fact that the function is not reversible (ie it is not possible to trace the

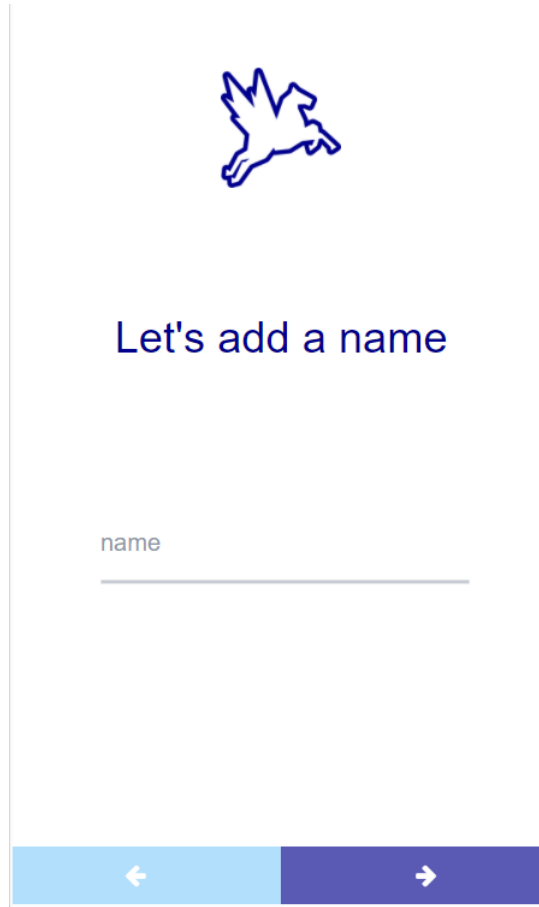


Figure 3.2. First step of the initialization process

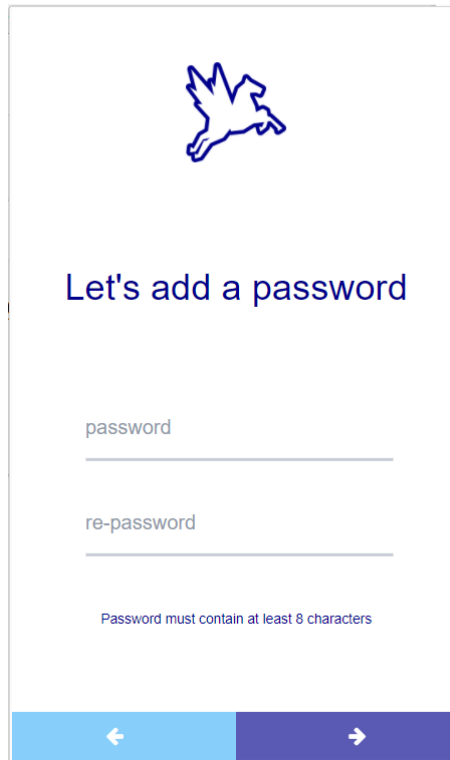
original message knowing only this data) and that it should never be possible to intentionally create two different messages with the same digest.

The last step in the registration process is the generation of the seed. It is more than just a password as it is the real key to access your funds on the wallet. The seed is an alphanumeric string consisting of the uppercase letters of the alphabet and the number 9. Unlike other cryptocurrencies, where the backup / private key are almost always automatically generated and stored by the wallet itself, in IOTA the Seed is totally independent from the purse, it must be generated and stored separately. the possible combinations to be used as IOTA seed are.

$$27^{81} = 8.710^{115} \quad (3.1)$$

In IOTA, there are 3 security levels to choose from. A security level determines the number of rounds for hashing, which means that a single seed can have 3 different accounts (Fig 3.4).

Private keys are derived from a seeds key index. From that private key it is possible to generate an address. The key index starting at 0 position and can be incremented to get a



The screenshot shows a mobile application interface for setting a password. At the top is a blue outline of a fox head. Below it, the text "Let's add a password" is displayed in blue. There are two input fields: the first is labeled "password" and the second is labeled "re-password". Below the second field, a small blue text note states "Password must contain at least 8 characters". At the bottom, there are two buttons: a light blue button with a left-pointing arrow and a dark blue button with a right-pointing arrow.

Figure 3.3. Second step of the initialization process

Security Level	Security
1	81-trits (low)
2	162-trits (medium)
3	243-trits (high)

Figure 3.4. Security level

new private key, and thus address. It is important to don't forget that all security functions are implemented client-side. What this means is that it is possible to generate private keys and addresses securely in the browser, or on an offline computer. All libraries provide this functionality. IOTA uses the one-time signatures of winternitz, so you need to know which private key has already been used in order not to reuse it. Subsequently reusing private keys can lead to the loss of funds (an attacker is able to forge the signature after continuous reuse). As such, never reuse private keys and addresses! If you are waiting for a transaction/bundle to confirm, always reattach it and never respend. Because IOTA uses Winternitz one-time signatures, you should never reuse an address after you have spent from it. Continuously reusing

private keys gives a sophisticated attacker the ability to forge the signatures, thus being able to steal funds from the respective address. In this application, the seed is generated according to the following procedure:

- a seed of 81 characters is generated randomly
- once this seed is generated, to reach a level of security greater than that already provided by the `crypto.js` module for the generation of this string, it is necessary to change at least 10 characters of this seed. To do this, simply click on the character relative to the one you want to change (Fig 3.5).
- once you have changed at least 10 characters, you can continue.

Crypto.js is a module in Node.js which deals with an algorithm that performs data encryption and decryption. This is used for security purpose like user authentication where storing the password in Database in the encrypted form. Moreover provides set of classes like cipher, hash, HMAC, decipher, sign, and verify. The instance of that class is used to create Encryption and Decryption. Node.js cannot create a class object using the new keyword. In this application, the `crypto.randomBytes()` function was used to generate the seed. This method will not complete until there is sufficient entropy available. This should normally never take longer than a few milliseconds. The only time when generating the random bytes may conceivably block for a longer period of time is right after boot, when the whole system is still low on entropy. Once the seed has been generated, it is necessary to export it as it was said before, in IOTA, the seed is independent of the wallet. It is possible to use the same seed on multiple wallets since the wallet offers only an access interface to the iota network, and the access key is this seed To do this a functionality has been implemented that offers the possibility to export the seed once generated correctly (Fig 3.6).

Once the seed was correctly generated, it was necessary to find a safe methodology for memorizing this seed as it represents the key to accessing its own funds. The seed is encrypted with the password not-hashed entered by the user and is also saved in the variable related to the session, so that, if the application is not used for more than one hour (time for which the session is not deleted) , the password is deleted, and, in case of attack, the hacker will not be able to decrypt the seed and thus steal the funds. Each time the user logs in, the password



Let's generate a seed

Press **10** more letters to randomise them

T F N B S T F A J
9 N I Q T A R X O
M L B R R X Z 9 B
X V 9 E P P S B 9
C A L O V C Y O C
V B O D D I T E 9
I A K T U C L A X
G I I S C 9 L I N
B L D G T J B R D



Figure 3.5. Seed generation

entered is stored in the session variable while the password hash is permanently stored in the local storage.

The AES-256 algorithm was used to encrypt the seed. Advanced Encryption Standard is the most popular symmetric encryption algorithm in the cryptographic world. The standard works on 128-bit block sizes and includes 128, 192 or 256-bit encrypted scripts (reported as AES-128, AES-192 and AES-256 respectively). The corresponding keys are equally extensive. Many coding solutions, such as TrueCrypt, have embraced AES from the beginning. Perhaps, however, the most significant fact in the success of the standard was its adoption by the US government in 2002 and the passage in 2003 to the format for the protection of classified data.



Let's export the seed

Take care to copy the seed in order to correctly
reinitialize the wallet

XFVJFBPWRQ9XLHAEISJEYOVBXOFY...



Copy to clipboard



Figure 3.6. export of the seed

The AES coding is based on a replacement and permutation network, which means that a series of mathematical operations creates highly modified (coded) data. Input is always simple text and the key is used to drive operations. These can be as simple as a single-bit rotation (bitwise rotation), XOR (exclusive OR) or more complex. Since a single pass should be simple to decipher, all modern coding technologies work on multiple occasions. The AES cycles are repeated for 10, 12 or 14 times for AES-128, AES-192 and AES-256. The AES keys also bear the same process as the user data, which is reversed in a changing key. The AES encoding process works with 4×4 addresses of individual bytes called boxes, S-boxes for replacement and P-boxes for permutation (they are separate stages). The replacement works inside boxes

while the permutation exchanges information between the boxes. S-box working on complex principles, which means that if only a single bit in entry is modified, the more output bits will be affected or that the properties of each single bit in output depends on each incoming bit. Repetition of the passages makes the codification good, as dissemination and confusion criteria must be respected. Diffusion occurs through the cascade combination of S-box and P-box transformation. By changing only a single bit in the input text, S-box will change the output of several bits, while P-box will semi-distribute the effect between different S-boxes. A minimum change in entry has a maximum impact on exit, a real avalanche. Once the initialization process has been completed, it will be possible to take advantage of the functionalities offered by the application. These features are accessible from the Home page (Fig 3.7). At this point in the application it is possible to monitor the amount of funds and check in detail all the incoming and outgoing transactions. In order to check in detail a transfer (bundle), it is necessary to click on the relevant line in the "Transactions" section. At the click of this line, the application will show the detail screen of this transfer each of which identified by:

- **date:** timestamp of confirmation of the transaction
- **type:** *received* or *sent*
- **state:** it can be *pending* or the transaction has not been confirmed or it can be *confirmed* if it is confirmed.
- **amount:** MIOTA number sent or received in this bundle

For each user, the application also creates a pair of keys (public and private) so that they can be used in the key exchange mechanism sent by the sensor to access the MAM channel on which it is writing the data collected. The exchange exchange meccansimo will be explained later. Since it is not possible to generate client-side sessions, this application implements a session manager in order to automatically log out if the application is not used for more than one hour. The implementation of this logic was developed thanks to the use of *Web Storage*. Every time a user logs in the application, the system writes a session variable to the local storage. This variable consists of date and time related to access. Every time the user reopens the application, in case of previous success during login, the system will check that the last access goes back less than an hour before the last access. If this interval is longer than one

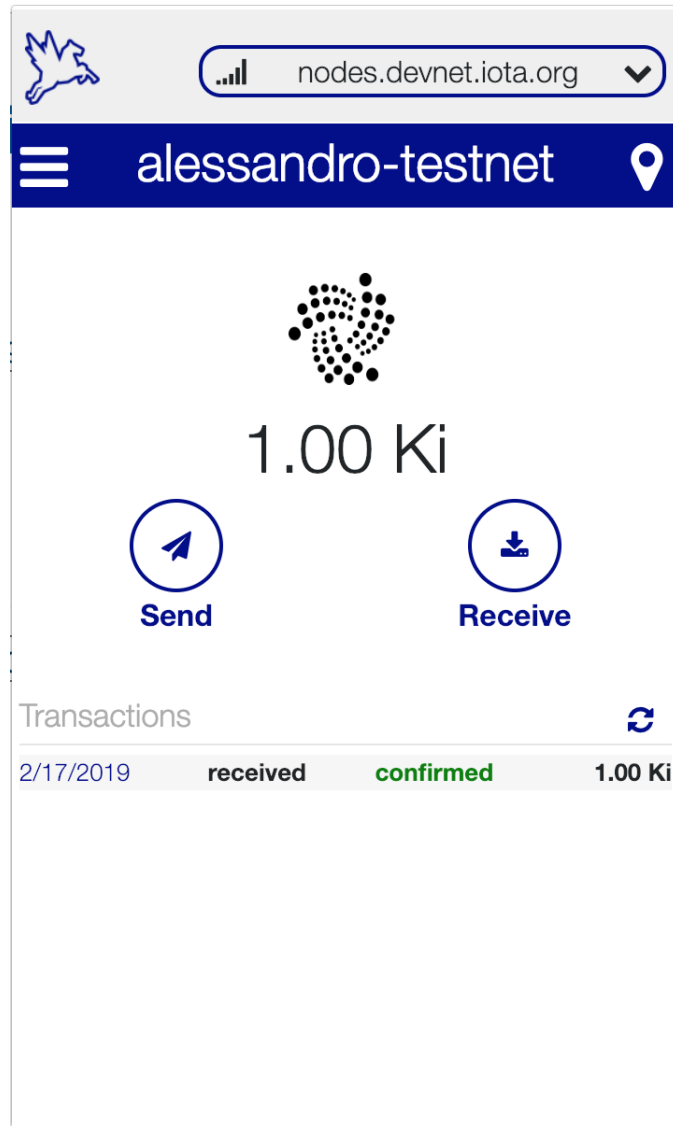


Figure 3.7. Home view

hour, the application will delete the session variable from the local storage and redirect the user to the login page. Among the historical limitations of JavaScript was the impossibility of storing data in a persistent manner. The only two approaches available used cookies for local persistence and the server for remote persistence. But while the first approach has limitations both in the amount of data that can be stored and in flexibility of access, the latter involves the presence of a server-side backend that takes care of the actual persistence and assumes to be online. Things have changed, also thanks to the advent of HTML5. JavaScript has a series of APIs that allow you to manage data more flexibly and without intermediaries. In this section, we will explore the solutions available to manage data persistence with JavaScript. Web Storage

can be considered the evolution of the cookie-based approach, by means of which a JavaScript application can store data locally on the browser. Unlike cookies, Web Storage provides more disk space, generally around 5MB, and information is never transferred to the server. However, as for cookies, the same original policy is maintained, according to which only the scripts that come from the same domain can access the same *Web Storage*. The Web Storage specification provides two objects for data persistence. Unlike the properties of a normal JavaScript object, however, the values that can be assigned to a storage property must be of string type. So if we are going to store complex data such as objects or arrays, we must first serialize them, for example in JSON.

Login

In case of logout or in case of cancellation of the session variable, the application redirects the user to the login page where he can enter the password or in case of password lossing, he can reinitialize the wallet by providing the seed as long as he possesses it. Every time the user enters a character, the application calculates the hash of the entered password and compares it with the hash stored in the local storage. In case the two hashes are equal, the login button will change from disabled to enabled so that the user can actually click it. Once correctly logged in, the application generates the session variable containing the time and password related to the access, to which it is used to decrypt the stored seed in the initialization phase in the case of a transfer of funds. Once this procedure is complete, the application shows the homepage screen (Fig. 3.7) and the user can access all the features offered by the application. As far as navigation within the application is concerned, it has been chosen to develop an ad-hoc component that implements a navigation bar. This component has been developed trying to make it as much as possible adaptable to all types of events that could be generated within an application.

Send

The following application allows the possibility to send both MIOTA tokens and the possibility to send data thanks to the IOTA protocol, as this protocol offers the possibility to send free transactions (Fig 3.9). To send an IOTA transaction, the sender's device simply checks two previous transactions in the Tangle, performing Proof of Work with very low difficulty. No transaction costs, no separation between miners and users. Since to send a transaction you must

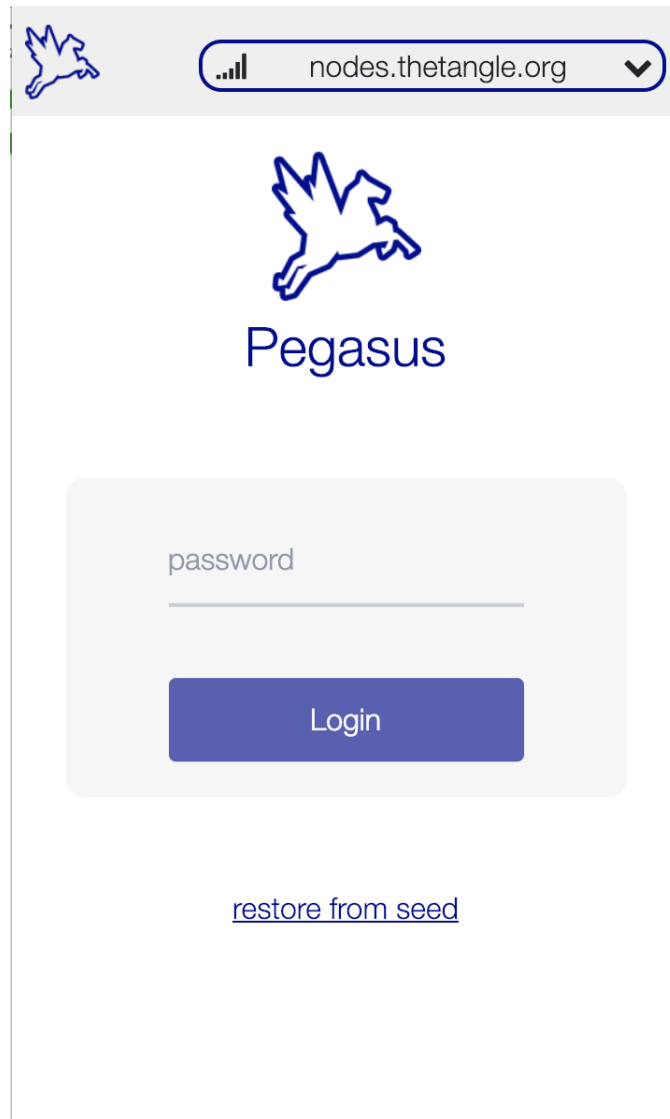


Figure 3.8. Login view

first confirm two more, as the number of users increases, the efficiency of the network also increases. IOTA scales proportionally to the number of transactions. Transaction execution times are inversely proportional to the number of transactions in the Tangle. When IOTA reaches mass adoption, the transactions will be almost instantaneous. An IOTA transaction consists of 2673 trytes. However, a user will usually broadcast a collection of multiple transactions (also called a bundle). One *bundle* could consist of withdrawal transaction (acquire funds to be spent from some address), payment transactions (pay some other address a sub or total amount of the funds withdrawn) or change transaction (deposit unspent funds back to one of the user's addresses). Bundles are not restricted to a specific number of transactions. For example, one

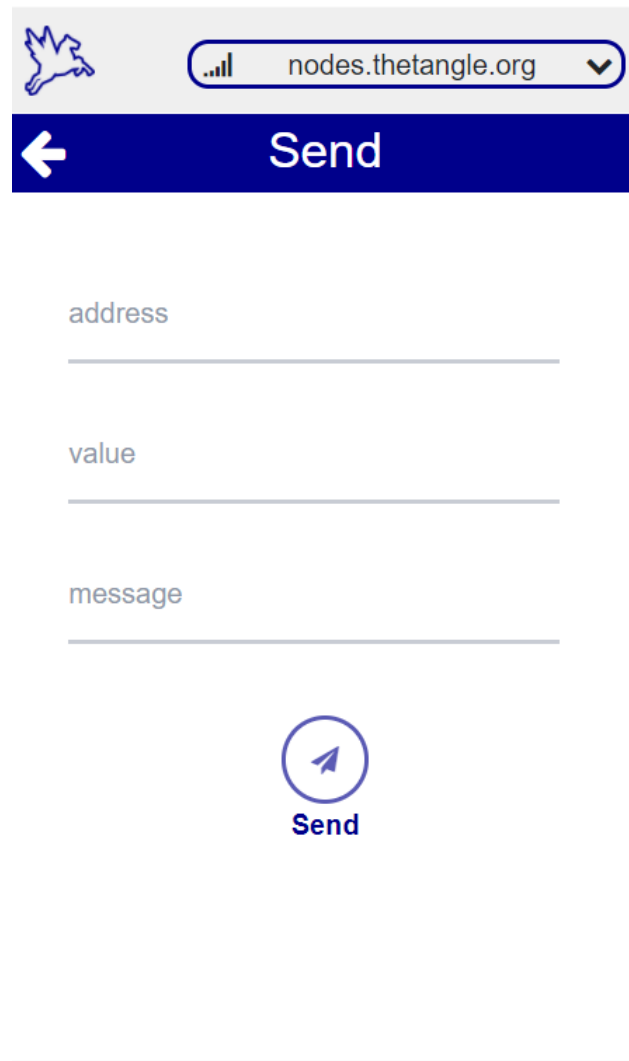


Figure 3.9. Send view

could have a bundle that consists of just one transaction (when a value of 0 is "transferred"). The data segments within one transaction are as follows.

- **Signature Message Fragment** contains the transaction signature = 2187 trytes
- **Address** is either receipt address (if value > 0) or withdrawal address (if value < 0) = 81 trytes
- **Value** is the amount of IOTA transferred = 27 trytes
- **Obsolete Tag** is some arbitrary user-defined value = 27 trytes
- **Time Stamp** of when the transaction was issued = 9 trytes

- **Current Index** of this transaction in its bundle = 9 trytes
- **Last Index** is the total number of transactions in the bundle = 9 trytes
- **Bundle Hash** is the hash of the entire bundle = 81 trytes
- **Trunk Hash** is the hash of the 1st transaction referenced/approved = 81 trytes
- **Branch Hash** is the hash of the 2nd transaction referenced/approved = 81 trytes
- **Tag** is another arbitrary user-defined value = 27 trytes
- **Attachment Time** is the time stamp for when PoW is completed = 9 trytes
- **Attachment Time (lower bound)** is a slot for future use = 9 trytes
- **Attachment Time (upper bound)** is a slot for future use = 9 trytes
- **Nonce** is the PoW nonce of the transaction = 27 trytes

Receive

Another feature offered by the application is that it can receive transactions. Since the receipt of transactions implies that the sender knows the address of the recipient, it is necessary to provide a mechanism for exporting the address in order to make it public. For this reason, the application offers the user the possibility to export his / her address via a QRcode or copy / paste it (Fig ref receive).

Bundle details

As mentioned previously, it is possible to consult the details of a bundle by clicking on the relevant row in the list of transactions on the home page. IOTA uses an account-like scheme. This means that we have inputs (addresses) which you have to spend in order to transfer tokens. Addresses are generated from private keys, which in turn are derived from a trytes-encoded seed. A transfer in IOTA is a *bundle* consisting of outputs and inputs. Bundles are atomic transfers, meaning that either all transactions inside the bundle will be accepted by the network, or none. A typical transfer in IOTA is a bundle consisting of 4 transactions. A unique feature of bundles is that the transactions are identified via the bundle hash, but also via the

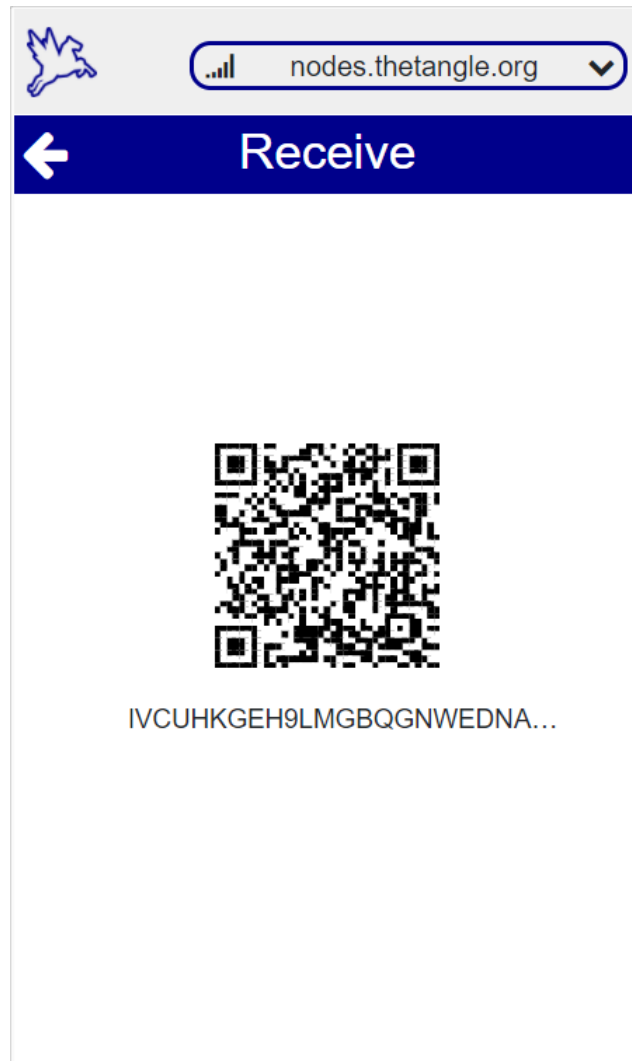


Figure 3.10. Receive view

trunkTransaction . What this means is that the tail transaction (*currentIndex*: 0), references in the *trunkTransaction* the transaction hash at index: 1 , *currentIndex* 1 transaction references (and approves) index 2 and so on. This makes it possible to get the full bundle of transactions from just a tail transaction by traversing down the trunk transaction. A single transaction can obviously contain multiple inputs and outputs. Once the application shows the bundle detail page, it is possible to consult the hash related to it and all the transaction hashes in the bundle. Since a bundle consists of several transactions, for each of them, the application shows its value (Fig 3.11). As you can see in Fig. 3.11 there are two buttons related to the functionality of *promoteTransaction* and *reattachTransaction*.

The behaviour of these two actions is different:

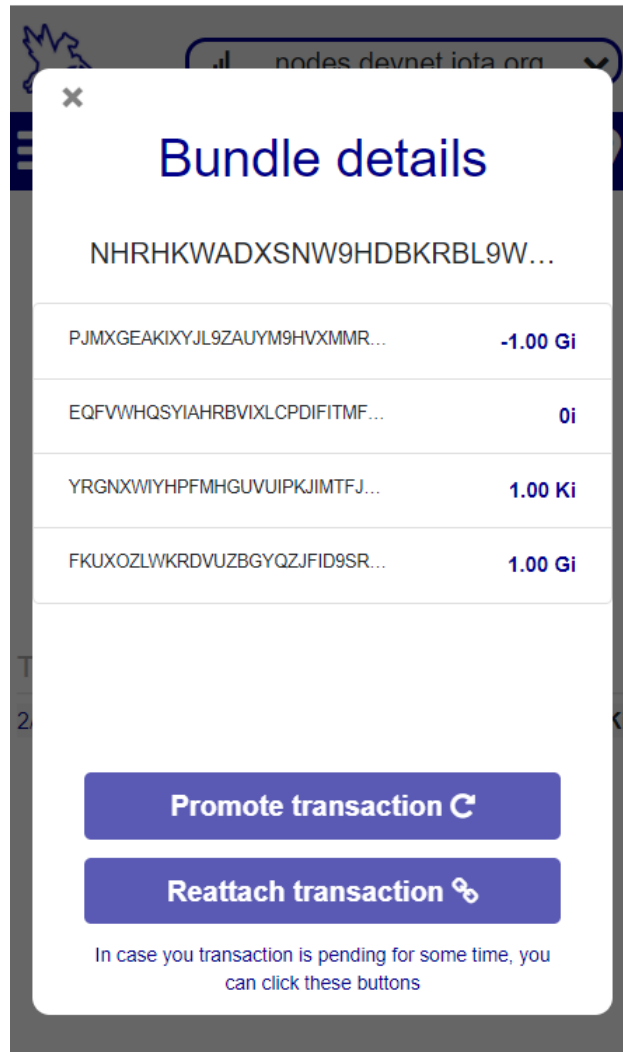


Figure 3.11. Bundle details of an incoming transaction of 1Ki

- **Reattach** will pick two new random tips and create a copy of your transaction bundle attached to them. This needs more PoW and basically creates a double-spend situation in which only either the original transaction or the reattached ones can confirm.
- **Promote**, on contrast, issues a single new 0-value transaction which will confirm your transaction and the latest milestone. In some cases, it will not directly approve them, but indirectly: It will run a MCMC walk from your transaction as well as a MCMC walk from the latest milestone, and the randomly chosen "tips" will be used as tips for the new transaction. That way, the weight of the new transaction will be increased and the chance of your tip of being chosen later is increased. Also, when you promote a transaction which

has already been promoted, your second promote will promote the first promote, reducing the number of tips.

In order to make the buttons enabled, it is necessary to check the status of the transaction, as these functions must not be available if the transaction has already been confirmed. In the IOTA javascript library, a function *checkConsistency* was added which can be called on a pending transaction and it will give you a feedback about the transaction. This feedback indicates if it is still possible for this transaction to be confirmed, or not. When the transaction is still consistent, you can only promote it, and if not, you can only reattach it.

Change network As the development of complex applications requires a test environment, it has been necessary to implement a function that allows you to easily change the network on which you are operating. This functionality has been implemented through the development of a drop-down menu containing a list of nodes to which it is possible to connect (Fig. network). In case a user wants to connect to the Mainnet, it is necessary to select the entry *nodes.theTangle.org*, while in case he wants to operate on the Testnet, it is necessary to select the entry *nodes.devnet.iota.org*.

Add/Modify Account

This application offers the possibility to manage multiple accounts, as a user may need to have more addresses on which they can receive or send payments. For this reason it was necessary to implement the functionality of adding an account (Fig 3.13). To access this feature, you need to open the side-bar menu using the appropriate button on the navigation bar. As a user could have multiple accounts, this case study had to be managed at the data structure level. each user is identified by an identified one calculated as follows:

$$id = sha256(accountName) \quad (3.2)$$

Once the button for adding an account has been clicked, the system shows the procedure related to this function, which is composed of three main steps: the addition of the name, the generation of the seed using the procedure described for the part of initialization of the wallet, and the

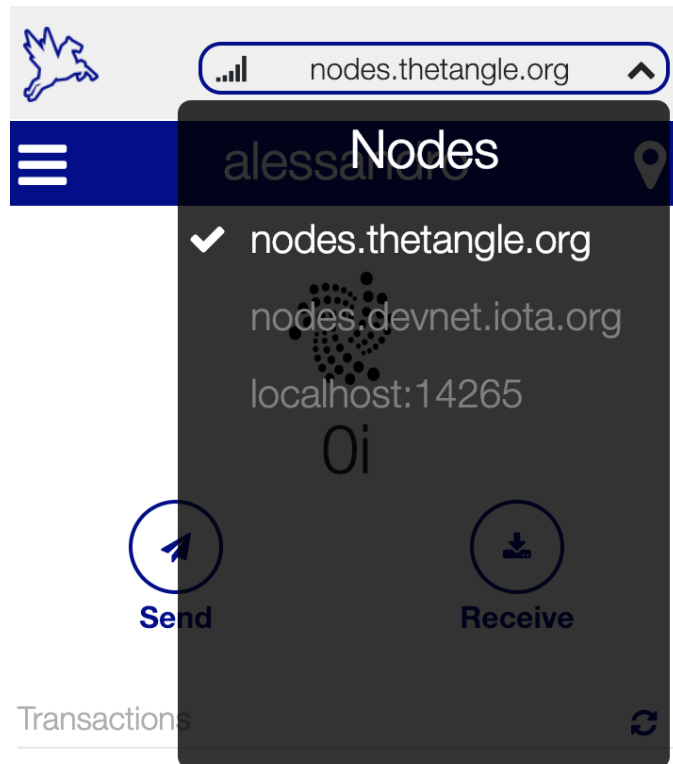


Figure 3.12. Drop down menu containing the nodes that can be reached

export of the newly generated seed. Since it is possible for a user to insert a name already in use, creating a situation in which there are two accounts with the same name inside the wallet, it was necessary to apply a control in order to avoid this situation. It is necessary to consider on which network (Mainnet or Testnet) the new account is being created. For this reason the basic data structure of the application has been divided into two parts: the part relating to the Mainnet and the part relating to Testnet, in order to don't create conflicts because the generation of an address is dependent on the network on which you are operating. The entire data structure is saved in the local storage in JSON format so you can easily modify it. it is possible to view the data structure of the application in Fig. 3.14 In addition, it was necessary to enter for each

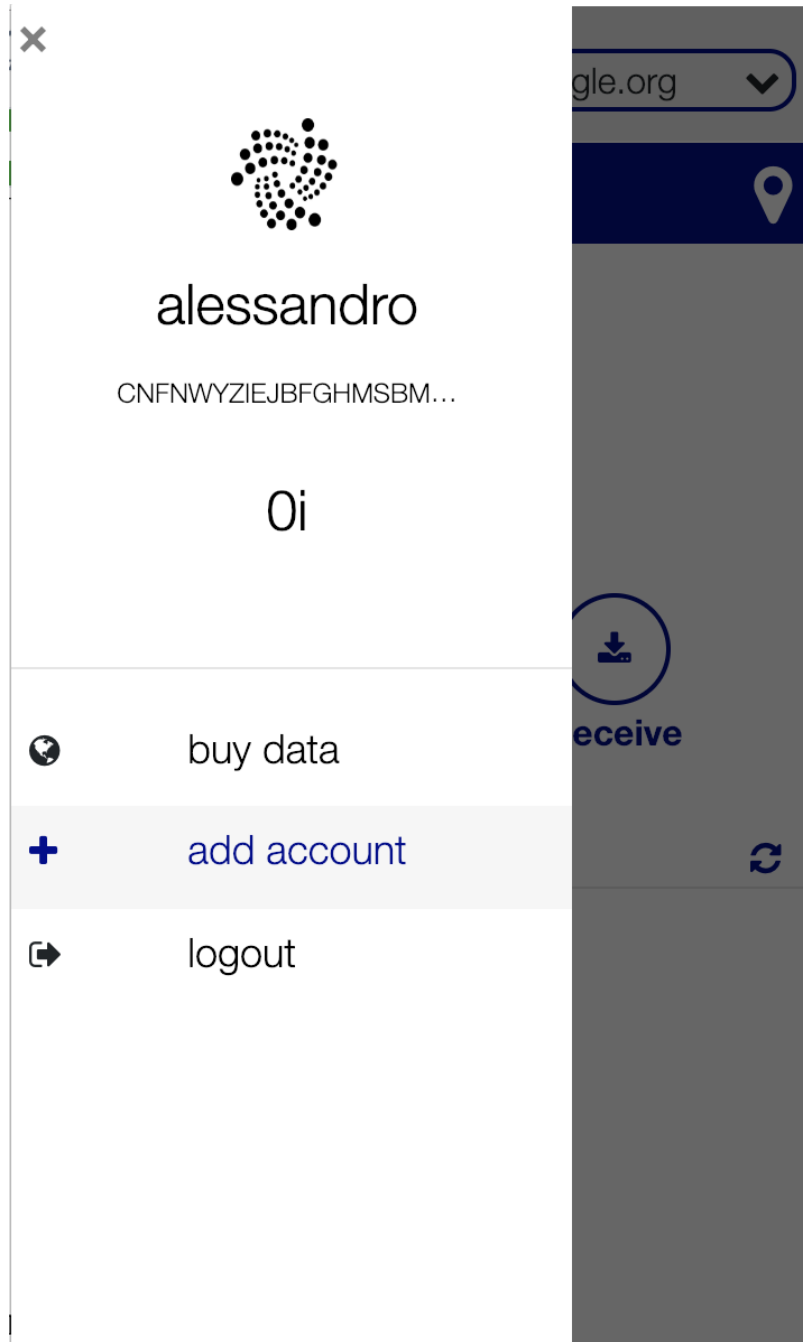


Figure 3.13. Add account

account a variable that allows you to keep the current account in use. This variable is called *current* and every time an account is created, the value of this variable is set to true in order to change the status of the current account in use. Since the data has been divided by network type in order to avoid conflicts, it is obvious that there will be two accounts belonging to different networks having the same value of this variable set to true. Since the application initialization

```

▼ {mainnet: [{name: "alessandro",...}], testnet: [{name: "alessandro-testnet",...}]}
▼ mainnet: [{name: "alessandro",...}]
▼ 0: {name: "alessandro",...}
  current: true
  ▼ data: {latestAddress: "FXMUKOZCJIBJUXLDMTSQIF9JDMMATJQHTWZDOXCAXXPVTFGZHHZJDACHFEBGXMGUKZORVOAVCXW9ZRWX",...}
    ► addresses: [{"CNPNWYIEJBFQHSBHMJUMPZLRGKMWWSQEDITOBMGDQQTQCAFY9QVLJDBCKSROZCRKYQTZSTGKOLHTAD",...}]
    balance: 0
    inputs: []
    latestAddress: "FXMUKOZCJIBJUXLDMTSQIF9JDMMATJQHTWZDOXCAXXPVTFGZHHZJDACHFEBGXMGUKZORVOAVCXW9ZRWX"
    ► transactions: [{"WISSOLITSKAWHRAVA9NBUYQMGKSYMTXDTGMGZOKKVVYZZGLQGBNQIXSKFAHKMIPFPIQRHDTNFVH99999"}]
    ► transfers: [{"hash": "WISSOLITSKAWHRAVA9NBUYQMGKSYMTXDTGMGZOKKVVYZZGLQGBNQIXSKFAHKMIPFPIQRHDTNFVH99999",...}]
    id: "8b555452b14c8d3570608964a1ce406429731746211d45dd3c92c21fb35cb039"
  ▼ marketplace: {keys: {privateKey: {type: "Buffer",...}, publicKey: {type: "Buffer",...}}, channels: []}
    channels: []
    ► keys: {privateKey: {type: "Buffer",...}, publicKey: {type: "Buffer",...}}
    name: "alessandro"
  ► network: {name: "nodes.thetangle.org", provider: "https://nodes.thetangle.org:443", id: 0, type: "mainnet"}
  seed: "2875f308084f626f4990dba516f8ca4e31f98aa52d770bed50ee5af8383275da584b2944890a7f254b91f9a68d139f19cfa5709cae487c9e302c1cf5d8ddab1ab0ac1eb9744a2d89138b18b2f24949a998"
▼ testnet: [{name: "alessandro-testnet",...}]
▼ 0: {name: "alessandro-testnet",...}
  current: true
  ▼ data: {latestAddress: "WAQNIJXJQNPDLRJHYNTLDIIAZYHQC9CPH9FWQFLDZV9MXCFTIGOZCTKMMATGJOKOHUYKYGPQMUF9HILC",...}
    ► addresses: [{"XBJUGKZHLHLPORPNHUEAOLKROXKZMQAFKCEMITIMFCTAGAVNGKPIXPDSXBKOTIARRTIKFBKYNFAOLSD",...}]
    balance: 1000
    ► inputs: [{address: "XBJUGKZHLHLPORPNHUEAOLKROXKZMQAFKCEMITIMFCTAGAVNGKPIXPDSXBKOTIARRTIKFBKYNFAOLSD",...}]
    latestAddress: "WAQNIJXJQNPDLRJHYNTLDIIAZYHQC9CPH9FWQFLDZV9MXCFTIGOZCTKMMATGJOKOHUYKYGPQMUF9HILC"
    ► transactions: [{"QARGLCIUFSLEBPJXZYXRFCKMKVHGKCCZZFANTJGRCDFEZHAKRZGFMCBRRGCCSRDEOQBXYTXCKD9999"}]
    ► transfers: [{"hash": "ARBMIJDSHTCZQMRUQMOGUPRULFSRSVZKF9SGNZBMCSPWLDJWKHUJHRLDPQNOVGRHZ9UEBWAZCHW999",...}]
    id: "9e74415fd58ebdb973ad7664684166d95f296584dda01f344c715541afee12f"
  ▼ marketplace: {keys: {privateKey: {type: "Buffer",...}, publicKey: {type: "Buffer",...}}, channels: []}
    channels: []
    ► keys: {privateKey: {type: "Buffer",...}, publicKey: {type: "Buffer",...}}
    name: "alessandro-testnet"
  ► network: {name: "nodes.devnet.iota.org", provider: "https://nodes.devnet.iota.org", id: 1, type: "testnet"}
    id: 1
    name: "nodes.devnet.iota.org"
    provider: "https://nodes.devnet.iota.org"
    type: "testnet"
    seed: "387ff70a0659646f5491c7a08f87da4c3af694a0396c10fe49e549ef362e6bc24543295d8317623e5d8fe7ad6f6029216c6ab77f3aa516d83382917e9d5dfb01ea1b516a37e5b2c8f708217b3e15057ad99"

```

Figure 3.14. Application data structure

procedure operates on the Mainnet, it is possible for a user to change networks without having yet created an account related to that network. In this case, the application creates a default account. Another feature offered is the ability to change the name of the current active account or to delete the account. This feature can be reached via the sidebar menu by clicking on the name of the account currently in use. At the click of this button the application shows the modal that allows you to change the name or cancel the account. (Fig. 3.16).

Restore account

3.3 Marketplace

The most important part of this thesis work is the development of a protocol above the IOTA network able to create a decentralized marketplace where payments are made thanks to the use of the MIOTA token. A decentralized marketplace allows for truly peer-to-peer transactions without centralized authorities taking their fees. This is made possible through IOTA technology. We no longer need the trusted third party verifying sellers and ensuring payments. Since the marketplace is decentralized, it was necessary to develop a protocol for which each connected sensor is able to supply its own data (eg: latitude, longitude, address, name, etc.). If the marketplace was centralized, this operation would have been simple as it would be enough

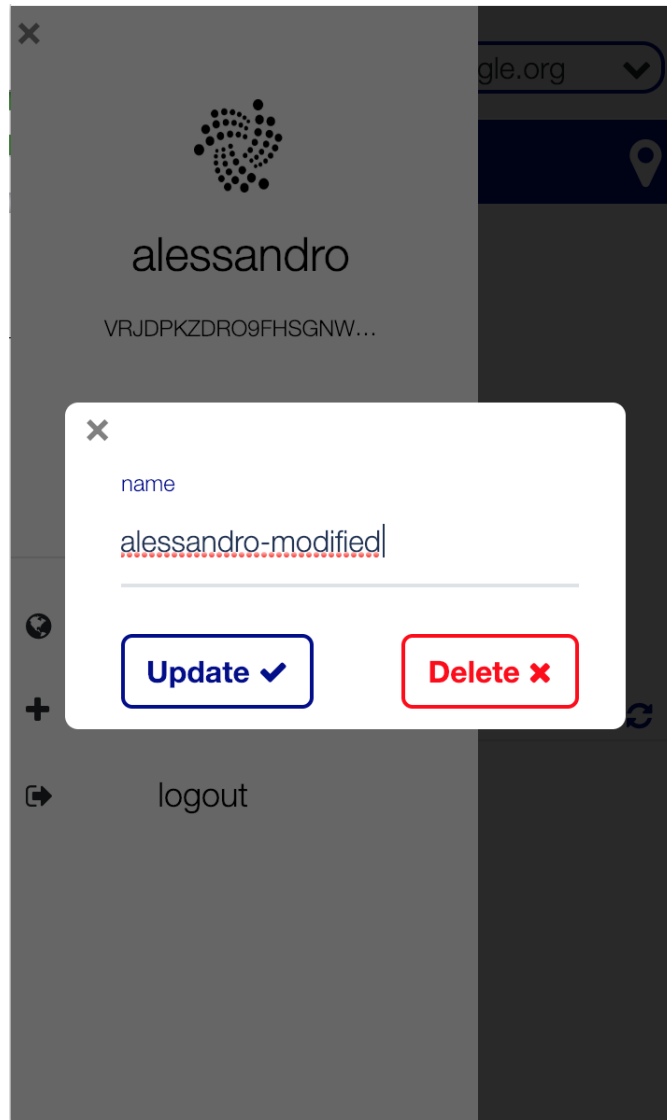
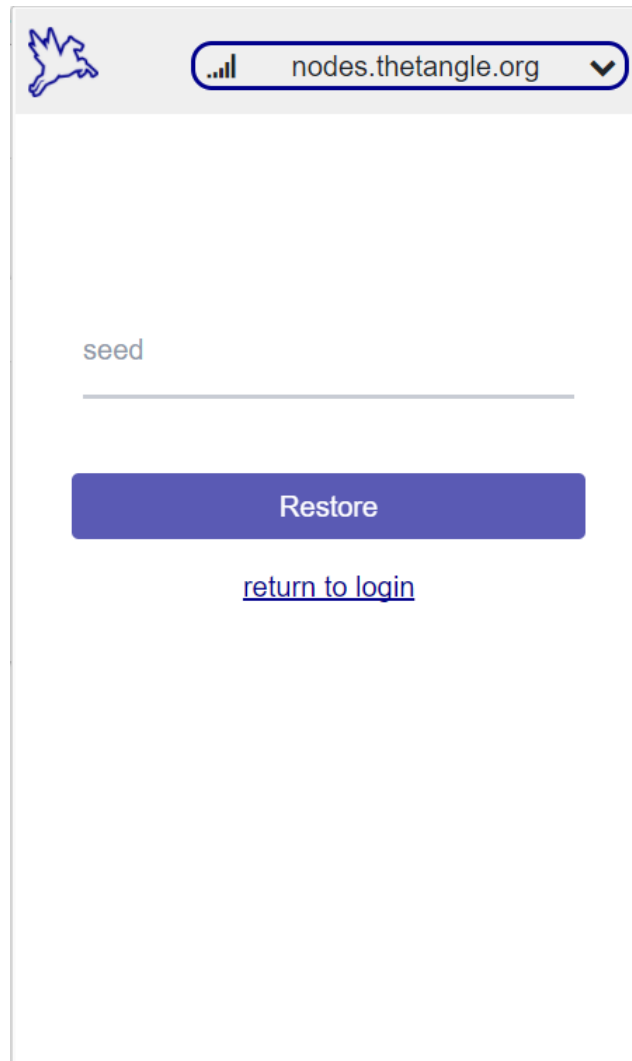


Figure 3.15. Modify an account name or deleting the corresponding account

to communicate such data to a web service, which will save them on the database and provide them to other users. In this case, the thing is more complicated: each sensor connected must communicate to all users connected their data without going through a server. To simulate this behavior, an ad-hoc application (PEGASUS-DAEMON) was created in order to simulate this scenario. The application, during the initialization process, in addition to initializing all the parameters necessary for the creation of a restricted MAM channel on which to send data, generates two seeds. The first one is related to the sensor account, while the other is used to simulate a receiver so that it is possible to send a transaction containing the sensor data to the receivers's seed. This transaction is identified by a specific *tag*, so that every user connected to



The screenshot shows a web browser window with the address bar displaying 'nodes.thetangle.org'. The page has a light gray background. At the top left is a blue Tangle logo. Below it, the word 'seed' is displayed above a horizontal input line. A large blue button with the text 'Restore' is centered below the input field. At the bottom, there is a blue link that says 'return to login'.

Figure 3.16. Restore account view

the marketplace is able to find on the Tangle all the transactions with this type of tag and then find all the sensors connected to the marketplace and their data. Once initialized, the sensor sends the detected data to the MAM channel every 20 seconds. Moreover, every time the sensor receives a payment from a user, it sends to this user the (symmetric) access key to the MAM channel and the root from which to start receiving data. Since the transactions sent on the IOTA Tangle are visible to everyone, it was necessary to implement a key exchange algorithm. For this purpose we chose to use the *RSA* algorithm. In cryptography, the acronym RSA indicates an asymmetric cryptographic algorithm, invented in 1977 by Ronald Rivest, Adi Shamir and Leonard Adleman, which can be used to encrypt or sign information. In 1976 Whitfield Diffie and Martin Hellman, American cryptologists, were the first to publish a system based

on the creation of an "asymmetric" cipher composed of "public keys"; although a few years earlier, James H. Ellis, Clifford Cocks, and Malcolm J. Williamson of the British secret services had already thought of it, the news was covered by military secrecy and was only disclosed in 1997. The cryptography system is based on the existence of two distinct keys, which are used to encrypt and decrypt. If the first key is used for encryption, the latter must necessarily be used for decryption and vice versa. The fundamental question is that, although the two keys are dependent on each other, it is not possible to go back to one another, so that even if one of the two keys is known, one can not go back to the other, ensuring in this way the integrity of cryptography. To create a public cryptographic system with an asymmetric cipher it is important that a user independently creates both keys, called "direct" and "inverse", and only publishes one. In this way we create a sort of "telephone directory" available to all users, which groups all the direct keys, while the inverse ones will be kept secret by the users who created them and used by them only when they receive an encrypted message with the respective public key of the "list" by a certain sender, thus obtaining the necessary conditions for the security of the system. To obtain a good security it is necessary to use binary keys of at least 2048 bits. Those with 512 bits can be obtained in a few hours. The 1024-bit keys, still widely used today, are no longer advisable. In fact, the factoring of large integers has progressed rapidly through the use of dedicated hardware, to the point that it may be possible to factor an entire 1024 bits in a single year, at a cost of one million dollars (a sustainable cost for any large organization, agency or intelligence). The marketplace is accessible through the application PEGASUS-APP, more

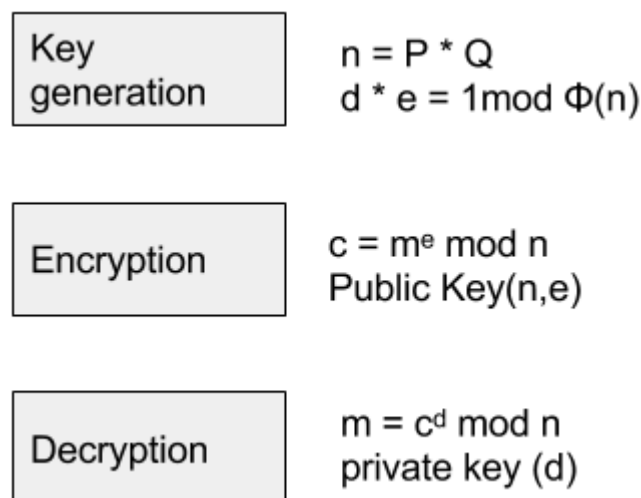


Figure 3.17. RSA

precisely, by clicking on the button on the right (marker-shaped button) in the navigation bar from the homepage. Once this button is clicked, it is possible to view all the sensors connected to the marketplace through a map. In order to implement this functionality Mapbox GL JS was used, which is a JavaScript library that uses WebGL to render interactive maps from vector tiles and Mapbox styles. It is part of the Mapbox GL ecosystem, which includes Mapbox Mobile, a compatible renderer written in C++ with bindings for desktop and mobile platforms. Every time a user accesses the marketplace, the application performs the following procedure:

1. Through the *findTransactionObjects* function provided by the *iota.js* library, the application searches for all transactions with the word "pegasus" as *tag*. In this way, as mentioned previously, it is possible to find all the devices connected to the marketplace. During this phase the application shows a loading screen (Fig 3.18).
2. To have the actual data of each sensor connected to the marketplace, for each transaction received, it is necessary to read the contents of the *signatureMessageFragment* field, and first through a conversion function from trytes to ascii, and then a conversion from ascii to JSON, it is possible to have access to the content of the data sent by the sensor, and then show the sensor-related marker on the map. The data sent by the sensor have the following structure:
 - *deviceName*: name of the sensor
 - *lat* : latitude of the sensor
 - *long* : longitude of the sensor
 - *address* : address of the sensor on which it is possible to make the payment to have access to the data stream
 - *price* : stream price
3. Once this data is correctly decoded, the application shows on the map the position relative to each sensor (Fig 3.19), thanks to the use of markers in order to facilitate the user. A marker is a visual representation of a specific coordinate on a map.
4. As a user may have purchased streams of sensor-related data, it is necessary to find all the transactions that identify a payment to a certain type of user by a given sensor, since

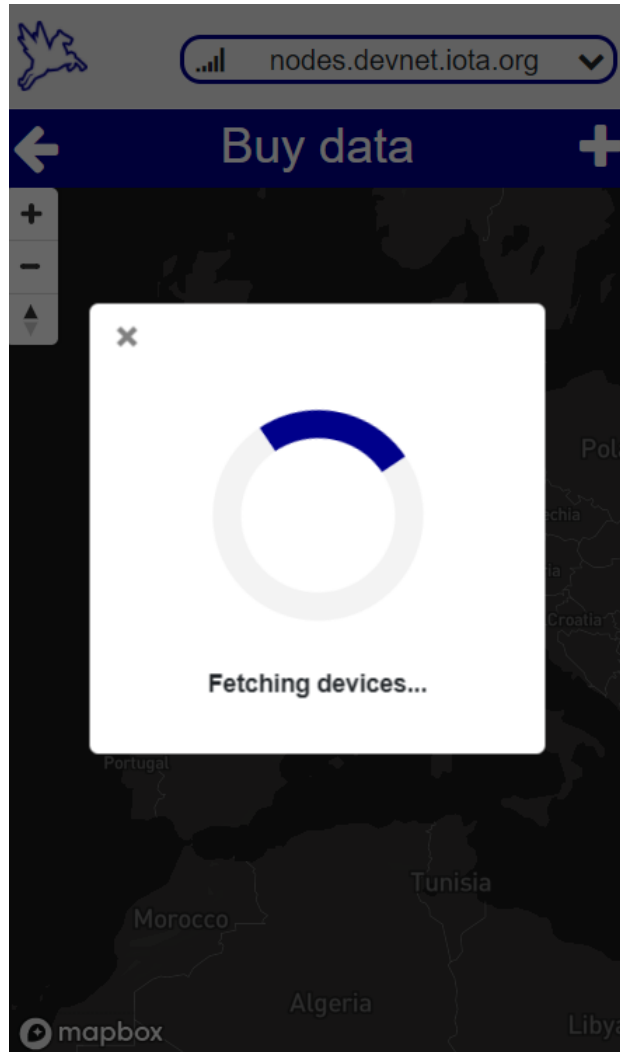


Figure 3.18. Loading device within the map

inside this transaction there are both the sidekey and the root to access the MAM channel. This type of transaction is identified by the "pegasus-payment" tag so that, thanks to the use of the *findTransactionObjects* function, it is able to find all the transactions issued by a sensor to a particular address. Again, once you have found all of them, you need to apply the decoding described above to the *signatureMessageFragment* field. The data sent by the sensor have the following structure:

- *root*: address from which to start fetching on the MAM channel
- *sidekey*: sidekey encrypted with the user public key sent during the payment.
- *deviceName*: name of the device

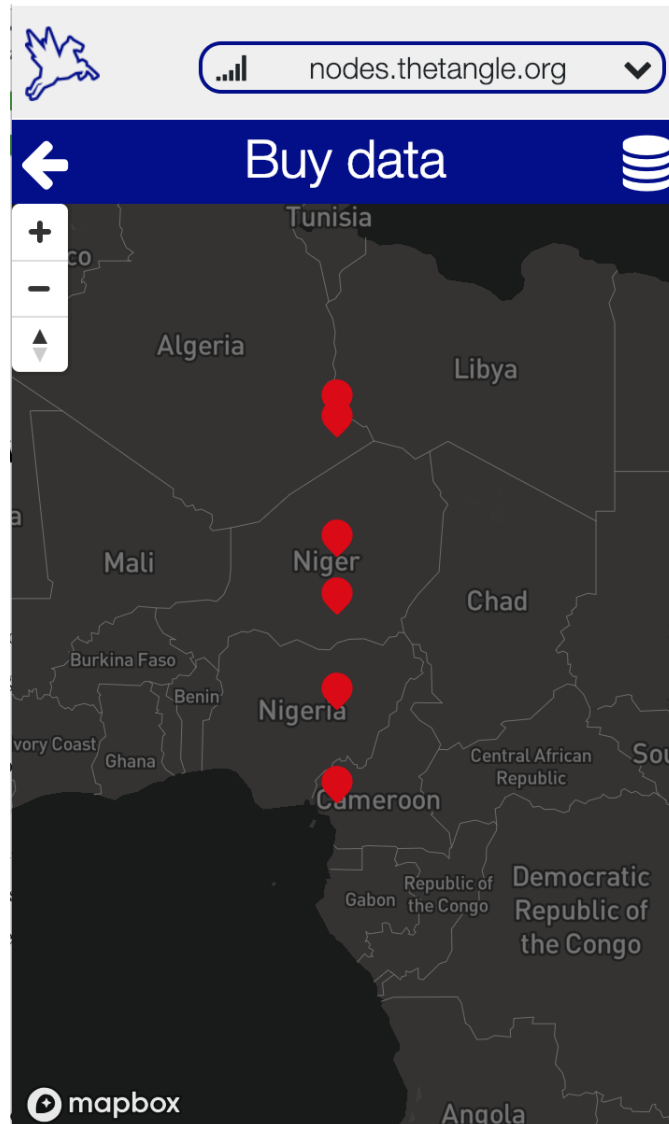


Figure 3.19. Sensor connected to the marketplace

5. Once the sidekey and root of each sensor have been received, the application executes a thread that every minute is able to receive the data sent by the sensors of which you have the root and key to access the MAM channel.

As for the payment by a user to a sensor, it is possible to access this functionality by clicking on the sensor marker on the map. Once clicked, a popup will appear containing the name and price of the selected sensor (Fig. 3.20). In addition to this information, there is a button that allows you to make the payment to the relevant sensor. By clicking on it, the application will send a payment transaction to the selected sensor. The amount of MIOTA to be sent is equal to the price shown in the popup. As mentioned previously, in order to access the sensor's MAM

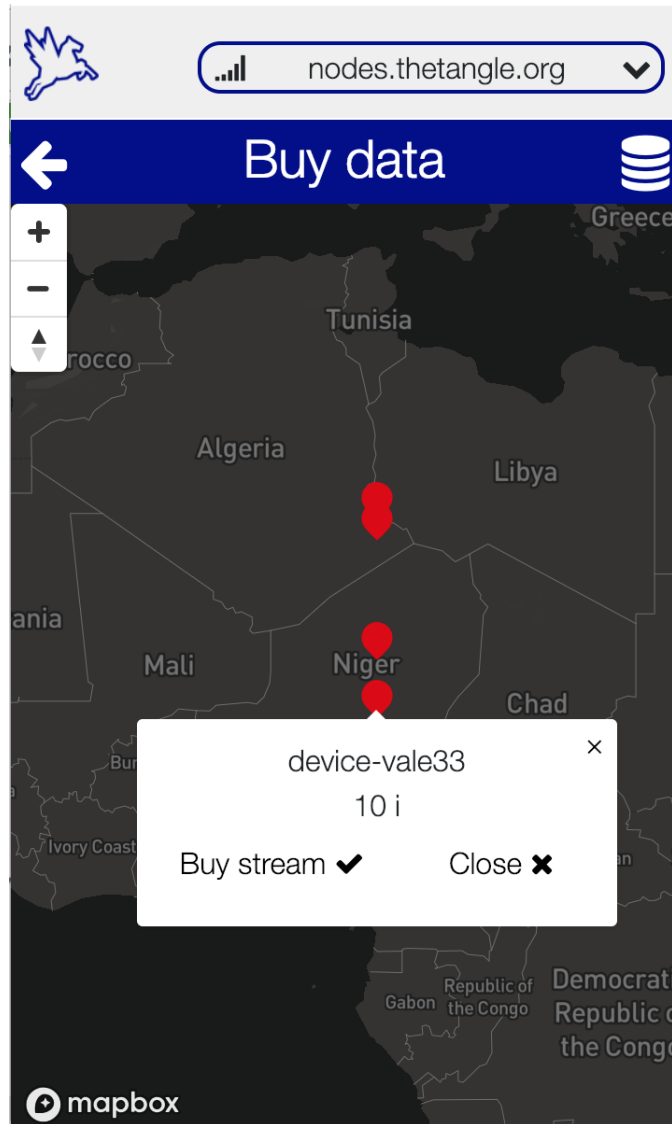


Figure 3.20. Buy data stream from device-vale33 for 10 iota

channel, it is necessary to have the access key and root. Since both the key and the root are sent to the user by the sensor once the payment has been made correctly, to avoid being visible to all users, as the transactions on the Tangle are public, these two parameters are encrypted with the public key of the user. For this reason during the payment, the application also sends this key to the sensor in order to allow it to correctly encrypt the data mentioned above. After purchasing data streams, the user must be able to monitor all the streams of data purchased. For this reason, a feature has been implemented that provides for the consultation of this data through a list containing all the names of the devices from which their data stream was purchased (Fig 3.22). To access this feature, you need to click the button on the right of the navigation bar in

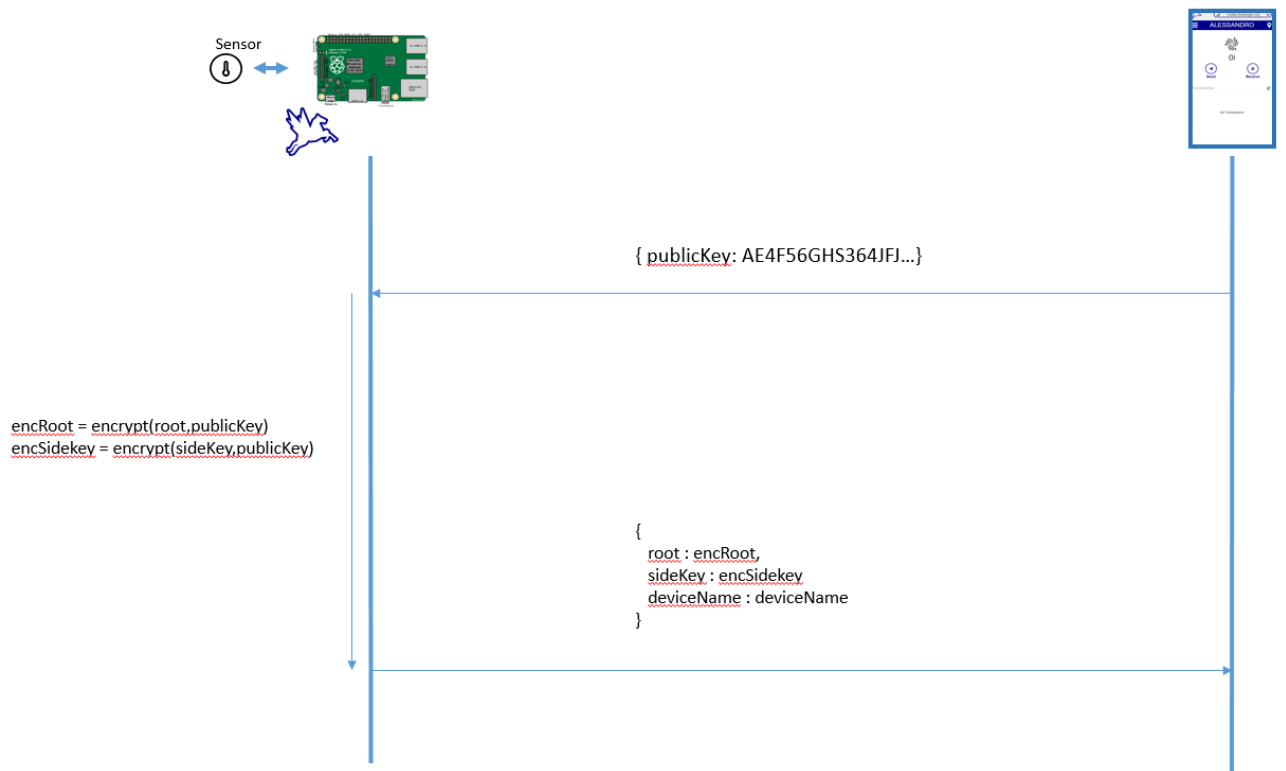


Figure 3.21. Payment process

the page on the marketplace. To access the data sent by the sensors, it is only necessary to click on the line relative to the sensor. For data reception, there is a process that takes these data every minute from the MAM channel of all the sensors whose access key and root is known.

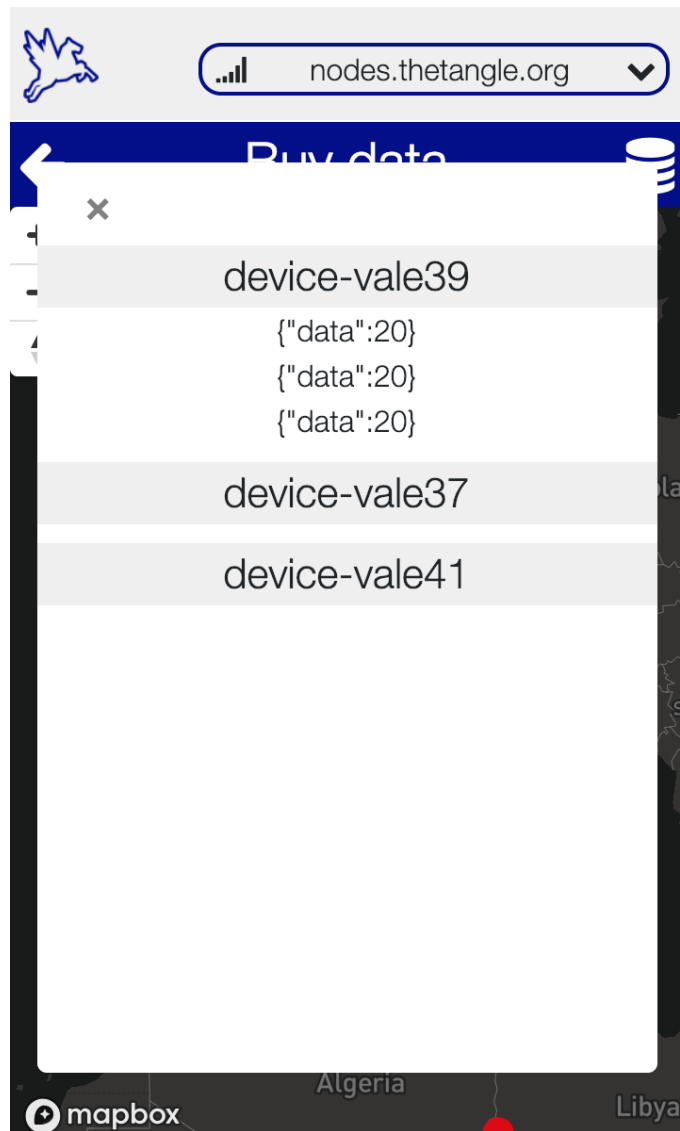


Figure 3.22. Purchased data stream

Chapter 4

Computational results

Having developed a very complex application, it was necessary to test its correct functioning in order to verify that all the functions described have been correctly implemented. Since the application allows operation on both Mainnet and Testnet, all functions have been tested on both networks. IOTA Foundation provides a series of tools in support of testing to ensure that developers can develop by testing the proper operation of the application. As the transactions are generated within the application, it is necessary to have a tool to monitor them. For this purpose, IOTA Foundation provides the *tangle explorer*, a platform that allows real-time monitoring of transactions and checking their status (Fig. 4.1). Through this platform it is possible to search for: wallet addresses, transaction hashes, bundles hashes or transaction tags. Moreover, it is also possible to monitor in real time all the transactions that are happening on the tangle and to know how much the TPS (transaction per second) corresponds at a given moment. As we have already mentioned, one of the main features of iota is the possibility to send transactions at value 0. for this reason, the platform allows you to choose whether to monitor all transactions or only those with a value of 0. Furthermore, it was necessary to have a similar tool to test the correct functioning of the data written by the sensors on the MAM channel. Through the tangle explorer, it is not possible to test this functionality as this platform does not allow to monitor MAM channels established on the tangle. Also for this purpose, IOTA foundation provides a platform that allows you to monitor MAM channels in which you know root and access key to the channel, in case this channel is restricted. This platform is called *mamexplorer*.

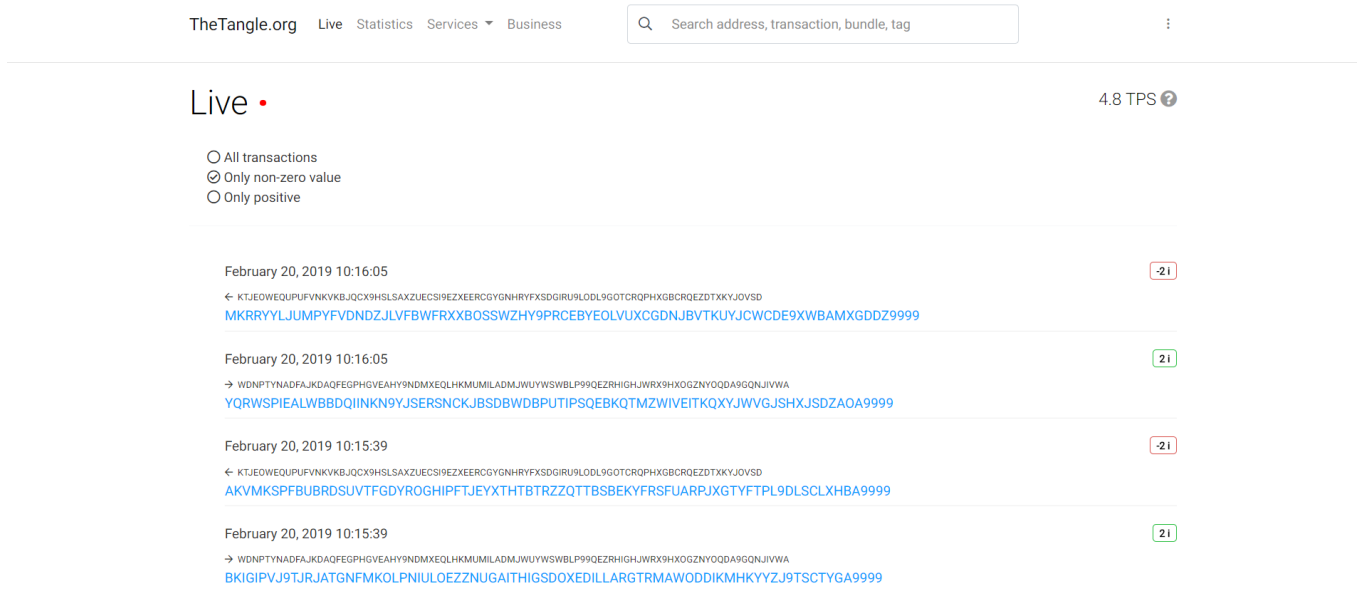


Figure 4.1. Tangle explorer

The screenshot shows the 'IOTA MAM Explorer' interface. It has a teal header with the IOTA logo and the text 'MAM Explorer'. Below the header, there's a form with three fields: 'Root *' (with a red error message 'This field is required'), 'Provider' (with the value 'https://nodes.devnet.iota.org:443'), and 'Mode' (with a dropdown arrow). Below the form is a large teal button labeled 'FETCH'.

Figure 4.2. Tangle explorer

Since in the application there is the possibility to send IOTA tokens, it is necessary to have within the wallet an amount of token for which it is possible to take advantage of this feature. For this reason IOTA foundation provides a service through which you can receive tokens on the test network. This service is called *faucet*. By entering the address of the own wallet in the appropriate form, it is possible to receive 1000 iota tokens.

4.1 Receive transaction

To simulate the reception of a transaction, the *faucet* service was used which, once inserted the address of the wallet on which the tokens have to be received, it generates a transaction at the address selected above with an amount equal to 1000 iota token. It is possible to check the status of the transaction thanks to the use of the tangle explorer (for the tesnet). The test for receiving a transaction consists of several parts:

1. Copy the address using the "receive" screen in the application (Fig. 4.3).

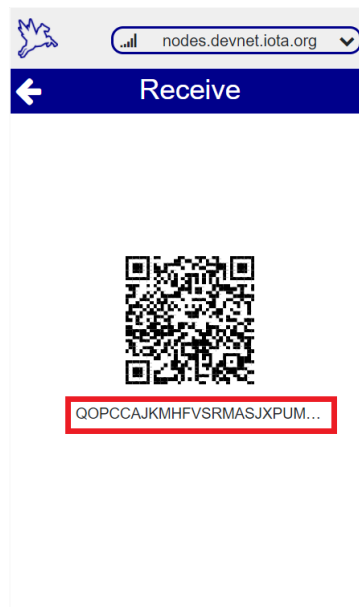


Figure 4.3. Copy wallet address

2. Paste the address in the appropriate form within the *faucet* platform and press enter (Fig. 4.4).
3. Wait for the wallet to synchronize with the tangle in order to be able to show this transaction (Fig. 4.5).
4. If the user wants to check the status of the transaction, just click on the row relating to the transaction whose status you want to know in the list of transactions on the home page and copy the address of the bundle or an internal transaction (Fig 4.6) to that bundle and paste it into the tangle explorer.

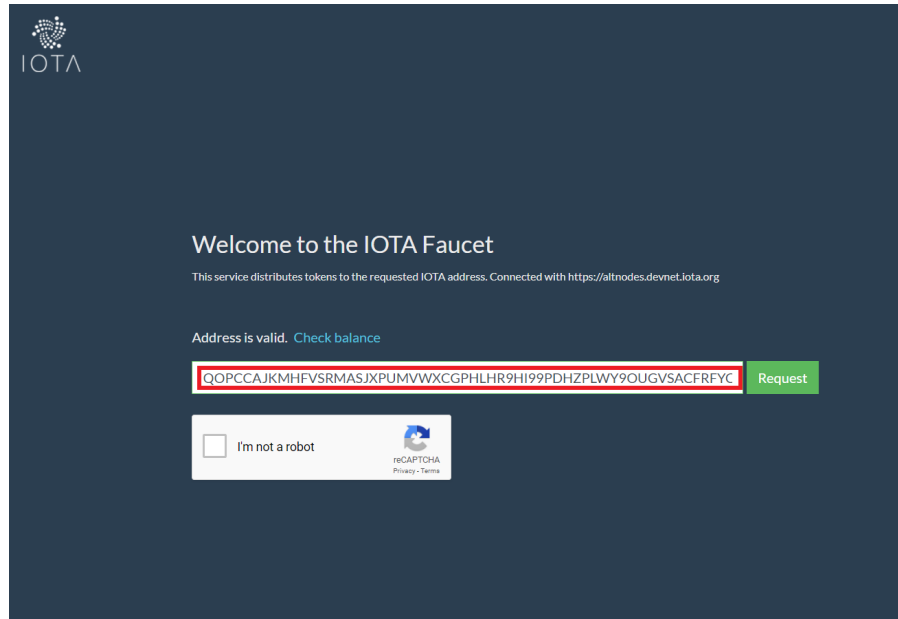


Figure 4.4. Paste wallet address within faucet form

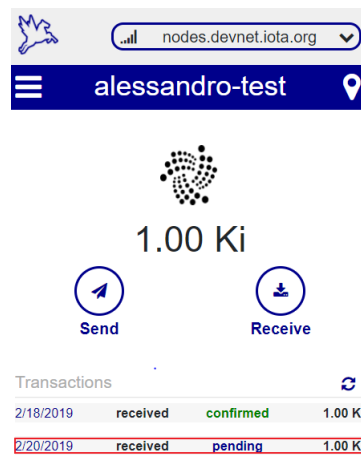


Figure 4.5. Paste wallet address within faucet form

4.2 Send transaction

In order to access the functionality of sending tokens, as well as having available a minimum amount for which you can send them, you need to know the address of the recipient. the application also offers the possibility of sending transactions at value 0 or entering 0 in the appropriate form relative to the amount of tokens to be sent, or leaving it empty. It is also

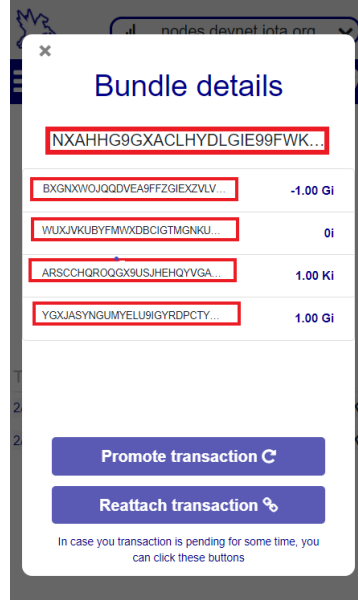


Figure 4.6. Bundle details

possible to send a message if the transaction is a value of 0 which is nothing but a text string that will be converted into trytes by the application at the time of sending.

4.3 Buy data stream from sensor

The main part of this thesis work was the development of a decentralized marketplace that allows a user to buy data directly from a sensor. The process of purchasing a sensor consists of the following steps:

- Access the marketplace through the application.
- Select the marker on the map relative to the sensor for which you want to purchase the data stream and click the "buy stream" button (Fig 4.7).
- Once clicked, it is necessary to wait for the operation to complete. In this process the application sends the payment and its public key to the sensor, so that it is able to send it the access key to the MAM channel on which it is sending data, encrypted with the public key just received (Fig 4.8). For technical reasons, such as connection problems related to the node with which you are interacting, it is possible that the process will fail. If the process fails, the message shown in Fig 4.9 will appear, otherwise, if the payment is successful, the application will show the message shown in Fig 4.10.

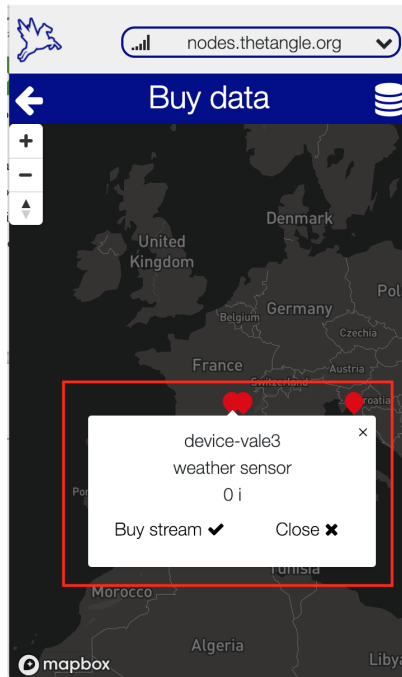


Figure 4.7. Selected sensor for which you want to purchase the data stream

```

start initialization ...
account seed : MAMQANIQXXAL9IBEUWSKHQIMJXIAQDT9ZONMENDBBYTTIGGEYFZVZKJ9TBBJKCY9PFZJX9YEMWFA9DYL
'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
start receiveing public keys
'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
device successfully communicate its detailsVKUWFN9C1JLFQJ99HJDJBZBARUEV9BENLOKHUVEFFYBURECHGXSROSKJFYH9ECTEBZUIGZYNUSOEBPIZ
Published { data: 20 }

'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
Published { data: 20 }

'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
Published { data: 20 }

start receiveing public keys
'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
start processing : QNEKNSPMVZYPUVWRVMEVXTQOEXGNPCCLLRQFYWZMXVKRZUJBKJ3SVUMHBIXSREFYCNSIWFVHDIRE99999
start processing : XWAQSKRRNZNEQPHPEZSKY9TGUHLBYUYDDMTAECEENLM09MWIEQXMRQPHCRRLUJLPLTGERGNPA9999
encrypted message
{
  sidekey:
    <Buffer 42 d2 93 6f c7 dd 85 2b 96 9f 90 59 eb 64 e0 61 cd e3 fd db 5b 3a 0a 64 3a 75 6e 97 b4 03 df 53 43 fd d6 e6 f1 50 a1 86 6c e9 71 6c e4 d4 46 42 c8 ... >,
    root:
      <Buffer 83 57 41 c3 ce ba d4 41 3b b3 3e cb be 3c 5f f3 d4 60 9c 3f 54 c2 96 31 56 d2 bf 34 e1 5c 4d aa 25 ac bb bd 69 2e 80 c8 a6 f5 a0 66 11 25 96 fd 25 19 ... >,
      deviceName: 'device-vale3' }
device successfully communicate send KRWKZEF5GLDYUVA9FZUQJEXSUVD0BANKIG0QZUVVEXGXWKEYSIAKIASIWDVWMD9WEJGWZFD0BQEAWA9PD at address ZHOR9RVSDXLJECBRIIFFLKRUUU9JTJXTUFCINIXRZMXDMNR
E4rHWGOSFAYOVFSRWVWVK9YPCFPCRZ
'AccountData.transfers' field is deprecated, and 'AccountData.transactions' field should be used instead.
Fetching of full bundles should be done lazily.
Published { data: 20 }

```

Figure 4.8. Message containing the sidekey and the root both encrypted

- In case the payment is successful, once all the transactions have been confirmed by the Tangle, it is possible to monitor the data that the sensors constantly write on the Tangle through the MAM protocol 4.11.

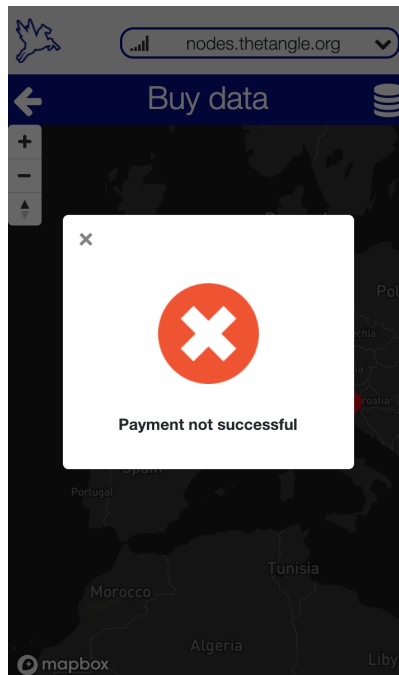


Figure 4.9. Error message during buying process

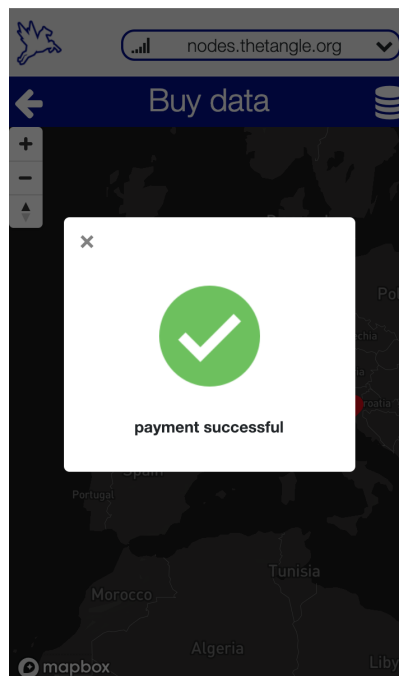


Figure 4.10. Success message during buying process

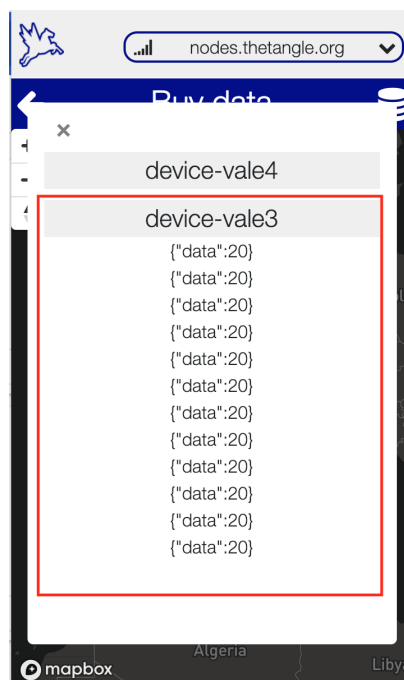


Figure 4.11. purchased data steam

Chapter 5

Conclusions

The rapid development of information technology profoundly changes the image of modern cities. To ensure a normal urban life, an infrastructure is needed that exploits technology to provide intelligent services. These technologies will become the basis for the construction of future mega cities. In 10 years, many infrastructural measures will be implemented. Smart Cities will therefore benefit from the introduction of a digital register technology such as IOTA. Combining sensors, smart contracts and high security in Tangle, the management of mega cities will be able to greatly improve the quality of life of citizens. The aim of this thesis work was to study the Blockchain and DLT, and to understand how these could be applied within a smart city. For this reason it was decided to study in depth the IOTA technology, that is a new generation cryptographic token, created to be lightweight and used in the Internet of Things. Since a smart city is based on the presence of sensors that independently produce data, it is necessary that at the base of it there is a layer that allows the management of them. For this reason, we have chosen to use the Tangle of IOTA as the basic layer for a smart city. The IOTA tangle has zero transaction costs, and it is infinitely scalable. This means that, it is not inhibited by the challenges of the Blockchain technology and it has the credentials to become the number one DLT for the development of smart cities all over the world. It is the perfect tool able to transform the world, solving urban-type problems to climate changes. The present thesis work, in addition to having analyzed which DLT is more suited to the concept of smart city, has also allowed the development of a PoC that allows a citizen to buy this data directly from the sensors installed inside the smart city, thus creating the concept of decentralized marketplace, as all these transactions take place on the Tangle of IOTA. With this type of

marketplace, every citizen is able to buy data streams directly from the sensors connected to the marketplace through the IOTA tangle without paying fees, as IOTA transactions are free. In addition, citizens are able to earn by installing sensors connected to the marketplace. For that reason Blockchain and DLT offer the possibility to create incentives to motivate the behaviour of citizens. Utilizing this functionality, cities or governments can achieve that citizens will make choices that are in the society's best interest. However, Blockchain and DLT are not without their challenges. The lack of coherent regulation has states and nations over-legislating to limit the scope of cryptocurrencies and Blockchain (and DLT), which can slow down transaction speeds and limit its access. Moreover, despite efforts at maintaining DLT's decentralized allure, there are outstanding issues of consensus building and universal adoption by users across the global network. The lack of a single, well-defined regulation between states and nations that supersede the laws to limit the extent of cryptocurrencies and Blockchain (and DLT), can slow down the speed of transactions and limit their access. Moreover, despite all the efforts made to maintain the decentralized fascination of the DLT, there are pending technical questions concerning the problem of consensus and how this type of technology could be adopted by all users in the world. In any case, it is expected it will require legal regulation. These regulations will have an influence on the future of DLT and Smart cities and as such could possibly disrupt the decentralized and free nature of blockchain and DLT by introducing a controlling, centralized participant such as a country.

Bibliography

- [1] D’Aliessi, M.: How does the blockchain work? medium.com (2016)
- [2] Foundation, I.: Iota docs (2018). URL <https://docs.iota.org/>
- [3] Popov, S.: Equilibria in the tangle (2017)