

Smart and lightweight cryptographic schemes to enhance trust and security in the Internet of Things

Kaningini Lutala Netho Junior (netho.j.k.lutala@aims-senegal.org)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Ado Adamou ABBA ARI, Ph.D., MBA
LI-PaRAD Lab, University of Versailles Saint-Quentin-en-Yvelines, France
LaRI Lab, University of Maroua, Cameroon

June 20, 2023

Master Thesis on Computer Security



Acknowledgements

I thank above all my God the Almighty, the creator of heaven and earth for giving me the breath of life and the power to realize my dream of doing the master's degree in a trustworthy institution like Aims-Senegal.

I would like to thank my thesis supervisor Ado Adamou ABBA ARI, Ph.D.,MBA for being always available and attentive, which allowed me to complete this thesis.

I would also like to thank my thesis co-director Michel Seck for his good advice, his good reflexes, his availability and especially for all the knowledge he transmitted to me.

Without forgetting to express all my thanks to Professor Franck Kalala Mutombo, who is also my Academic Director, for his advice and especially for his availability for the various questions related to the implementation of this thesis.

I also thank all the members of the administrative committee of AIMS-SENEGAL, Professor Mouhamed Moustapha Fall in particular, the President of the centre.

I thank my dear parents: Bernard Luatala and Marie Lutala, my brothers and sisters Jérémie Lutala, John Lutala, Demu Lutala, Godélive Lutala, Pauline Lutala and Agathe Lutala, to my very dear daughters Thérèse Lutala and Plamédie Lutala, to all my family for their unfailing support and continuous encouragement.

I address all my gratitude to Professors Pierre Kasengedia Motumbe, Pierre Kafunda, Jean Batubenga, and assistant Clement Amisi, all professors and assistant at the University of Kinshasa (DRC), for their encouragement and willingness to participate in the elaboration of this thesis.

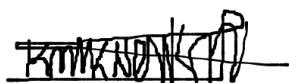
I cannot end without thanking all my comrades Arnaud Watusadisi, Ben Benteke, Cyrille Kyesiku, Saliou Fall, Aram Thiam, Mame Bator, Idriss Nguepi, Yvan Tamdjo, Engelbert Tshinde, etc... for their encouragement.

Abstract

The great power of the IoT is based on the fact that its objects communicate, analyze, process and manage data autonomously and without any human intervention. With the advent of smart homes, smart cities and everything intelligent, the Internet of Things (IoT) has emerged as a field with incredible impact, potential and growth. According to Gartner, 6.4 billion objects are already connected in 2016, and there should be 20 billion by 2020. For Etisalat, this number would rise to 28 billion by the same date, while according to Cisco, the number would be 50 billion in 2020. However, most of these IoT devices are easy to hack and compromise. In general, these IoT devices are limited in terms of computing capacity, storage and energy, and are therefore more vulnerable to attack than other devices such as smartphones, tablets or computers. The issue of privacy security is a topical one in the Internet of Things. In this work, we present and discuss the main security issues of the IoT. We review and categorize the main security issues of IoT. We describe the security requirements for IoT as well as attacks, threats, and present state-of-the-art solutions. Most importantly, our main solution will be based on the use of connected objects in a blockchain network. The blockchain allows us to have a distributed peer-to-peer network where untrusted members can interact with each other without trusted intermediaries in a verifiable manner. We're looking at how this mechanism works and we're also looking at smart contracts, the scripts that reside in the blockchain and allow for the automation of multi-step processes. We will describe how the blockchain - IoT combination can be used. This combination allows us to facilitate the sharing of services and resources, which leads to the creation of a service market between devices and also allows us to automate several time-consuming data shares in a cryptographically and verifiable way.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.



KANINGINI LUTALA Netho Junior, 07 January 2021

Contents

Abstract	ii
Introduction	1
Background of the study	1
Problem statement	2
Research Objective and Contributions	2
Organisation of the work	2
1 Generality of the Internet of Things	4
Introduction	4
1.1 Definitions	4
1.2 Life cycle of a Connected Object in the IoT	5
1.3 Composition of an IoT system	6
1.4 Functions of the connected object	7
1.5 Application areas of IoT	9
1.6 IoT Security	12
Conclusion	16
2 Cryptographic tools used in the IoT	17
Introduction	17
2.1 Generality of cryptography	17
2.2 Cryptography, data protection in the IoT environment	32
2.3 Gap in research	35
Conclusion	35
3 Research Methodology	37
Introduction	37
3.1 Blockchain for IoT Security and Privacy	37
3.2 Proposed architecture	44

3.3	Elliptic curve-based cryptographic protocol for secure communication and privacy security	49
4	Ethereum Blockchain	54
4.1	Mining Algorithm in Ethereum	55
4.2	Genesis Block in Ethereum	55
4.3	Ethereum Network	56
4.4	Sequence diagram of the different operations in the proposed architecture	60
	Conclusion	63
5	Implementation of the architecture IoT-Blockchain proposed	64
5.1	Blockchain-IoT Simulation	65
5.2	Data storage in the IPFS	68
5.3	Use of Infura and IPFS on the Ethereum blockchain	69
	Conclusion	75
	References	79

List of Figures

1.1	Life cycle of a device in the IoT	6
1.2	Composition of an IoT system	7
1.3	functions of the connected object	8
2.1	Schema of a cryptosystem	19
2.2	Symmetric Key Encryption	22
2.3	Data Encryption Standard (DES) Algorithm (Mushtaq et al., 2017).	24
2.4	Encryption and Decryption process of AES (Mushtaq et al., 2017).	26
2.5	Schema of Asymmetric Encryption	27
2.6	Hash function	30
3.1	Example of the linked blocks	38
3.2	Three-layer and Five-layer architecture of IoT	40
3.3	Five-layer architecture of Blockchain	41
3.4	Proposed architecture	44
3.5	Raspberry associated with an external storage disk(framboiseaupotager.blogspot, 2020)	46
3.6	IPFS's Structure	47
3.7	Blockchain Merkle Tree Root	48
3.8	Architecture of the Blockchain-IoT platform based on layers	49
3.9	Diffie-Hellman key exchange protocol	51
4.1	smart contract	59
4.2	Sequence diagram of the different operations	60
4.3	Sequence diagram of Identity for blockchain network user	61
4.4	Data sharing on IPFS by owner	62
4.5	Access to the data stored in IPFS	62
5.1	Python code for simulation Blockchain-IoT	66
5.2	Blockchain	67
5.3	Smart contract written in solidity	71

5.4	ABI and Address of the contract	71
5.5	Different layers of the application	72
5.6	User's Interface	73
5.7	IPFS's Transaction on Ethereum blockchain	74
5.8	The file stored on IPFS	75

List of Tables

- 2.1 Cesar encryption 21
- 2.2 Comparison between ECC and RSA (NIST parameters) 34
- 4.1 Differences between Bitcoin and Ethereum 54

Introduction

IoT technology consists of networks of devices that are connected, embedded, equipped with sensors and software that allow them to transfer data. This new field of innovation is gradually redefining a wide range of different sectors. The ability to collect data in real time brings a number of benefits to companies, enabling them to automate processes, boost productivity and improve the customer experience. The number of IoT devices in action continues to grow, with Gartner forecasting a record 20.4 billion connected devices in use by 2020. However, one of the biggest challenges for people working with IoT technology is security, with each IoT device acting as a potential entry point for hackers. In the event of a breach, sensitive information could be leaked on a massive scale or IoT devices could be exposed to the risk of hijacking by hackers. This is what happened in 2016 when the infamous Mirai botnet blocked access to many well-known sites, including Twitter, Airbnb and Netflix, by harnessing the power of approximately 100,000 IoT devices to launch a distributed denial of service (DDoS) attack. The answer to the security problem of the Internet of Things may well be blockchain technology. This transformation technology is a type of shared registry, also known as a distributed registry, and is probably best known as the basis of the Bitcoin crypto-money. Blockchain technology automatically stores data in multiple locations, rather than keeping it in a central repository, which makes it more secure. Using blockchain technology as the basis for IoT devices reduces the risk of hacking by reducing potential entry points and also ensures the security of privacy in the IoT network. By foregoing a central authority in the networks of the Internet of Things, blockchain technology could enable these networks to protect themselves. IoT devices in a common group could automatically stop working or alert the user if asked to perform any task that seems suspicious because it is outside of their usual remit. This would significantly reduce the risk of hackers hijacking IoT devices. With the reduction of potential entry points for hackers, this combination also provides an unassailable, tamper-proof logbook that makes it easy to track anything in the chain. Through the use of encryption and distributed storage, humans cannot overwrite registers. This kind of transparency also makes blockchain technology the ideal solution for intelligent contracts where an agreement can be automatically executed as soon as certain conditions are met.

Background of the study

IoT technology innovations provide remote monitoring, control, automation and status control of a wide range of devices and sensors that can be used in a variety of sectors including smart homes, autonomous cars, e-commerce, etc... Currently, the number of projects targeting the Internet of Things industry is increasing and privacy security is still an issue. The goal of our research is to increase the belief of the users of these devices by offering a solution related to the use of the connected devices in a blockchain network.

Problem statement

Nowadays, connected objects are part of our daily life. Cameras, lighting, sensors, there are several billion connected objects in the world that emit huge amounts of data every day. In order to function properly, connected objects use a special connection called the Internet of Things. This connectivity uses open external networks, on dedicated public networks or company networks connected to the outside world. A very large number of these connections are wireless and are therefore subject to attacks from the outside. The main risk of the connected objects is therefore the unauthorized penetration within this inter connectivity of embedded systems. It is enough for the malicious person to have access in the network, he can steal data, he can stop some services. If the object is therefore insecure, acts of hacking can easily succeed. The connected object sector remains a potential danger to the privacy and security of its users. In addition to potential cyber attacks, these everyday objects are subject to the commercial use of personal data and privacy breaches. Indeed, we leave many digital traces when creating online accounts and using connected objects. These data are centralized in data centers and are the target of marketing companies that freely analyze our personal data and lifestyle habits.

Research Objective and Contributions

We have proposed a solution that is based on the combination of the Internet of Things and the blockchain. Knowing that the blockchain is a technology for storing and transmitting information, transparent, secure, and operating without a central control unit. For the secure storage of data in the blockchain, we had associated the interplanetary file system (IPFS) technology that stores the data and returns its hash to be stored in the blockchain.

Organisation of the work

Except the introduction and conclusion, our work is divided as follows :

1. **Generality of the Internet of Things** : In this chapter, we have briefly explained the Internet of Things, its infrastructure and organizational aspects. We have reviewed some of the security threats within the IoT network;
2. **Cryptographic tools used in the IoT** : In this chapter, we have gone through the generality of cryptography passing from traditional cryptography to modern cryptography;
3. **Research Methodology** : In this chapter, we have proposed the architecture that combines the Internet of Things and Blockchain technology. We also presented the elliptic curve cryptographic protocols to guarantee confidentiality in data transmission in the IoT network;
4. **Ethereum Blockchain** : In this chapter, we explained the Ethereum blockchain which is a network of open and decentralized blockchains that also produces a cryptomony: Ether;

5. **Implementation of the architecture IoT-Blockchain proposed** : In this chapter, we have had to do some simulations of the operation of our proposed system.

1. Generality of the Internet of Things

Introduction

The concept IoT is used a lot currently due to its exponential rise in success. In recent years, not a day has passed without new information related to its reach in companies or commercial activities. The aim is to deploy to “things”, that are devices or software entities, the functionalities offered by the Internet in the field of communication in order to allow them to exchange between themselves or with humans, all kinds of information or data. The concept of connected objects is not new. According to the International Electrotechnical Commission (IEC), cash dispensers, which appeared in 1974, were the first connected objects (Ahmad and Ranise, 2018). Others consider that the notion of IoT was born at Carnegie Mellon University in the United States in the early 1980s, when students had the idea of equipping their department’s Coca Cola machine with sensors that could remotely monitor the number of bottles remaining in the machine and thus prevent a traumatic stock-out (CByothe.fr, 2020). Soon enough, as the Internet developed, similar ideas spread, often focusing on monitoring refrigerators and managing their contents. Kevin Ashton, co-founder of the Auto-ID Center at MIT, claims to have been the first to use the term “Internet of Things” to describe a network of communicating sensors, in a presentation to Procter and Gamble in 1999. Others consider that it was Steve Jobs’ announcement in 2007 of the launch of the iPhone that really marked the start of the concept, all previous achievements having remained anecdotal. It is true that it is since 2009-2010 that publications have multiplied and the most enthusiastic predictions have been made.

1.1 Definitions

Before defining the IoT, it is advisable to list some of the most remarkable components that explain its existence. In its simplest form, the IoT can be seen as a network of physical elements enabled by : Identifiers: to identify the source of data (e.g. sensors, devices); Sensors: to collect information; Internet connection: to communicate and notify; Software: to analyze data. We can say that the IoT is the network of things, with clear identification of components, integrated with software intelligence, sensors and ubiquitous connectivity to the Internet. The IoT allows things or objects to share information with the manufacturer, operator and/or other connected devices using the telecommunications infrastructure of the Internet (Hamidouche et al., 2018). According to the ITU (International Telecommunication Union), the Internet of Things is defined as “a global infrastructure for the information society, providing advanced services by interconnecting objects (physical or virtual) using existing or evolving interoperable information and communication technologies” (Boukara, 2019). Each object is uniquely identifiable through its integrated computer system and is capable of interoperating within the existing Internet infrastructure. Companies and/or technical analysts do not disagree that the number of things in the IoT will be massive. According to Gartner, 20 billion devices will be in use in 2020 (H.Khemissa, 2015), although this estimate can be revised in any case because of the pandemic (Corona) that the year 2020 is undergoing because according to Gartner’s projections, this could happen if the year

2020 does not experience the worst global imbalance in all sectors. Cisco estimated the number of devices (including machine-to-machine devices, telephones, televisions, PCs, tablets and other connected devices) at 26.3 billion for the same period (Rayes and Salam, 2017). According to the International Telecommunication Union ¹, the IoT is a “global infrastructure for the Information Society, providing advanced services by interconnecting objects (physical or virtual) through existing or evolving interoperable information and communication technologies”. The IoT, from a conceptual point of view, characterizes connected physical objects with their own digital identity and capable of sharing information with each other. This paradigm creates a kind of bridge between the physical and virtual worlds. From a technical point of view, the IoT consists of the direct and standardized digital identification (IP address, smtp, http protocols...) of a physical object through a wireless communication system that can be a RFID chip, Bluetooth or Wi-Fi.

1.2 Life cycle of a Connected Object in the IoT

Several companies today consider the IoT as a vector of transformation, to create new offers, increase operational excellence, or meet new regulations. However, the industrialization and scaling of IoT projects, which can involve up to several billion connected objects, means that it is necessary to anticipate the life cycle management of connected objects in order to guarantee the durability of the solutions implemented. Maintaining operational and security conditions is all the more complex as the volumes and diversity of connected objects are high, as any scenario can occur from the moment when fleets of billions of connected objects are considered. The life of an Object begins when it is manufactured. Due to the different areas of application (lighting, security, etc.), nodes are adapted to a specific task. It is therefore unlikely that a single manufacturer will manufacture all nodes for the same structure. Therefore, interoperability as well as trust-boosting between nodes from different vendors is important (Heer et al., 2011). In the IoT, Intelligent Objects go through three stages: the preparatory phase, the operational phase and the maintenance phase (Heer et al., 2011).

¹Youcef Baghdadi et al., ICT for an Inclusive World, p.498

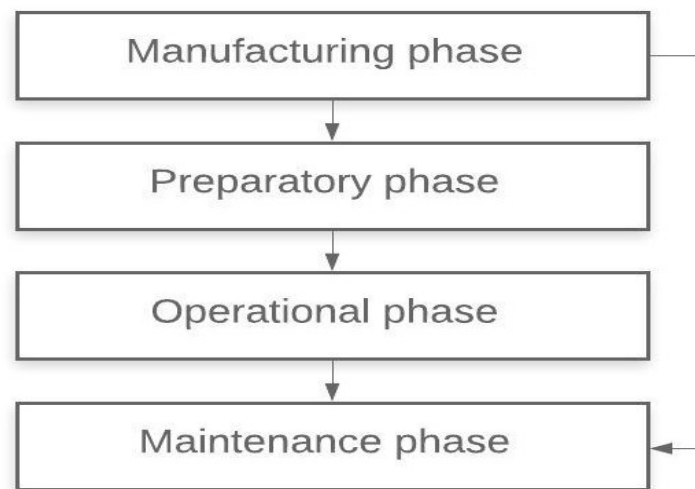


Figure 1.1: Life cycle of a device in the IoT

1. **The preparatory phase (bootstrapping)** : deployment of the Objects (sensors, tags), their configuration with the necessary information, for example identifiers, security keys, etc.
2. **The operational phase** : the connected Object starts to carry out its mission, which differs from one application to another.
3. **The maintenance phase** : performs updates, resolves problems by repairing the Objects in case of failures for example. It is even possible to replace Objects and restart again from the preparatory phase.

1.3 Composition of an IoT system

An Internet of Things system is composed of several levels that communicate with each other to connect the tangible world of objects to the virtual world of networks and the cloud. Not all projects adopt a formally identical architecture, however it is possible to schematize the data path (Connectwave, 2020). The system of the Internet of Things combines many actors and technological components. It is composed of connected objects, wireless communication networks, data collection platforms (hosting, processing), applications/services for end users and supervision/security of the entire chain.

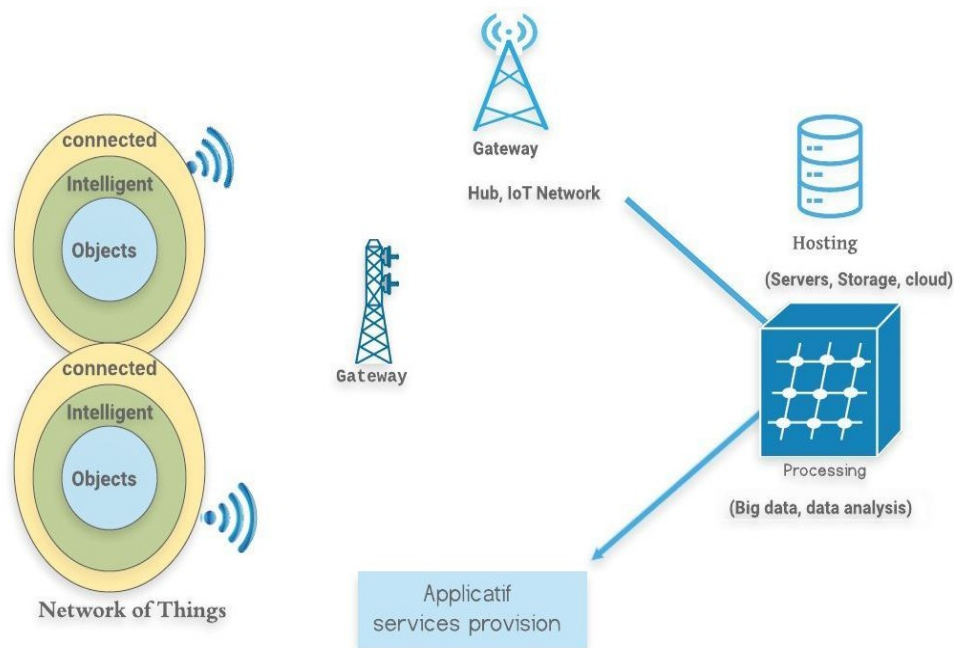


Figure 1.2: Composition of an IoT system

An IoT Gateway bridges IoT devices, sensors, equipment, systems and the Cloud to facilitate their communication. By systematically connecting “the field” with the Cloud, IoT physical gateways provide local solutions for processing and storage, and allow autonomous control of field devices based on data recorded by sensors.

1.4 Functions of the connected object

The connected object is first of all an object that has its own mechanical and/or electrical function, it can either be designed directly “connectable”, or it is already existing and the connectivity is added afterwards. The function of the connected object is to collect data from sensors, to process this data and communicate it using a connectivity function and to receive instructions to perform an action. Usually these functions of the connected object require a power source, especially when the data is pre-processed directly in the object.

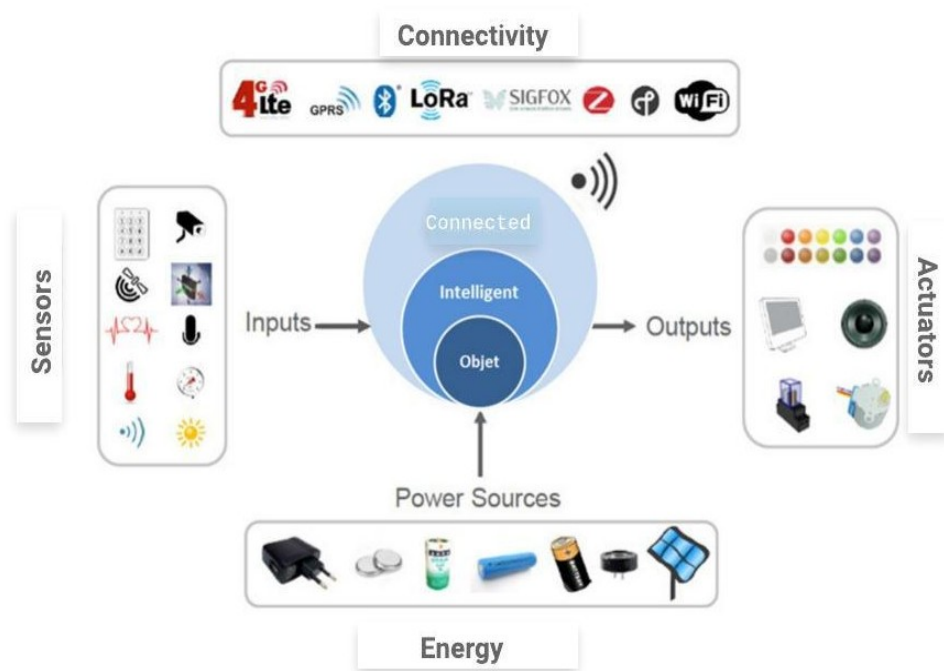


Figure 1.3: functions of the connected object

1.4.1 Sensors.

Sensors are devices that transform an observed physical quantity (temperature, luminosity, movement etc...) into a digital quantity that can be used by software. There is a very large variety of sensors of all types, the connected objects often have the function of capturing these physical magnitudes at their place of use. Example of sensors: light, presence, proximity, position, displacement, acceleration, rotation, temperature, humidity, sound, vibration, electrical, magnetic, chemical, gas, flow, force, pressure, level, ...

1.4.2 Energy sources.

There are 4 types of energy sources: wired power supply for objects with access to a power outlet, batteries for those who do not have access to it or only occasionally (recharging), energy sensors or "energy harvesting" (photo-voltaic, piezoelectric, thermoelectric, kinetic...) to extend the life of objects with very low consumption, and finally passive objects without batteries which are powered by the electromagnetic waves of the readers (RFID, NFC...). Energy is one of the great challenges of connected objects, both to guarantee the longest possible lifespan without maintenance, and to guarantee environmental respect despite the multiplication of energy-consuming connected objects that invade our personal and professional spaces.

1.4.3 Actuators.

Actuators are devices that transform digital data into a physical phenomenon to create an action, they are in a way the opposite of the sensor. Example of actuators: Displays, Alarms, Cameras, Speakers, Switches, Lamps, Motors, Pumps, Locks, Valves, Fans, Jacks, ...

1.4.4 Connectivity.

The connectivity of the object is ensured by a small Radio Frequency antenna which allows the communication of the object towards one or more networks ([Arouna Ndam Njoya and Gueroui, 2020](#)). Objects will be able to transmit information such as their identity, status, an alert or sensor data, and receive information such as action commands and data. The connectivity module also allows to manage the “object life cycle”, i.e., authentication and registration in the network, commissioning, updating and deletion of the object from the network.

1.5 Application areas of IoT

The potential of connected objects is enormous ([Ado Adamou Abba Ari, 2019](#); [Lemoine, 2019](#); [LesJeudis, 2020](#)). A 2016 study by Gartner predicts that by 2020, more than half of all business tools and processes will use the Internet of Things. The applications are varied and cover many fields: industry, science, health, etc.

1.5.1 IoT in the health field.

The current trend in health care in many countries today is to reduce a number of available hospital resources by relocating some health services to the home ([LesJeudis, 2020](#)). For example, medical check-ups are one such service. With connected health care, more work is needed to develop algorithms and models to use data for diagnostic and treatment decision-making activities. Connected objects are used on a daily basis for:

- Monitoring within medical establishments and maintenance;
- Surgical operations and remote control;
- Geo-localization services.

The standardization of the IoT in Healthcare will enable the creation of new business models that will increase employee productivity, but also collaboration between medical staff and communication with patients.

1.5.2 The digital revolution in response to energy imperatives.

Artificial intelligence is a real added value in the field of energy, which could in the coming years represent a decisive ecological investment in the future of our society. The stakes are also economic, and companies have understood this. IoT, in the context of energy, responds to major issues:

- depletion of natural resources;
- increasing global energy needs;
- market price volatility;
- shortage of manpower.

The digital revolution has entered the debate in the energy sectors through the management of resources: energy meters, smart grid but also the Internet of Things in the home, with the smart home.

1.5.3 Home automation or connected home (smart home).

One of the most outstanding examples of IoT is the smart house. Users operate mainly in the home network environment. Nowadays we can find many intelligent objects in our homes. These smart objects are adapted to our needs. In addition to entertainment objects such as smart TVs or connected speakers, home automation has also thought about safety and energy saving in the home :

- Home automation control unit: control and programming of various interventions inside the home;
- Information sensors (alarm system, temperature variations, etc.);
- Actuators, which allow the programming and control of the various electronic devices in the home, even remotely.

1.5.4 The connected industry.

The industry is not outdone in its use of the Internet of Things and the benefits it brings. Within the framework of the problems encountered in the industrial field, the use of connected objects is very specific and meets needs:

- Optimization (logistics);
- for business process transformation;
- improving efficiency and productivity;
- traceability and security.

The digital revolution is also an opportunity for certain types of industry to renew themselves and bring added value to a field that is losing popularity.

1.5.5 The Internet of Things in Agriculture.

The rapid growth of the world's population, changes in alimentary habits and climatic disturbances are three major factors, among others, that make modern agriculture a daily challenge. By 2050, agricultural productivity will need to have increased by 70 %to meet global demand. More than a technological challenge, this is a humanitarian issue. Cereal and market gardeners have already taken advantage of drones to collect information in real time that is essential for farm management:

- soil moisture;
- planting situation;
- climate, etc.

The collected data is transferred to the connected (sometimes autonomous) tractors. This makes it possible to finely dose the level of fertilizer and watering on a given plot of land and to reduce costs, both financial and energy.

1.5.6 The connected objects in the livestock.

The connected objects are not only useful for farmers, but also for stockbreeders who can monitor the health of their animals more closely. Have you ever heard of connected cows? They are the most connected animal in the world! Its collar equipped with numerous sensors allows better traceability but also real-time information on its health and behaviors.

1.5.7 Smart retail: trendy supermarkets.

Physical commerce is also undergoing the transformations of the digital era. Strongly challenged by e-commerce and m-commerce, retail shops want to take advantage of the popularity of IoT by combining e-commerce with traditional sales. Physical shops have also taken the lead in the digital revolution and are increasingly offering fun and interactive features to enhance the sales experience and increase conversion rates. One of the concepts of “smart retail” is Radio Frequency Identification (RFID) technology, which enhances the customer experience by offering a highly personalized customer journey. In addition to mobile applications, concepts of connected shopping carts have already been developed to facilitate supermarket shopping:

- integrated shopping list;
- guided tour to optimize the race time;
- automatic calculation of the basket amount,...

Merchants are also investing in mobile applications to retain and attract customers to physical stores, for example, through notifications on promotions/sales.

1.5.8 Smart and connected cities.

Called “smart cities” or “connected cities”, cities have begun their digital transition to meet the challenges of modern society. Urban planning, the economy and sustainable development are important issues for the cities of tomorrow, which must meet the needs of an increasingly dense population with increasingly limited resources. Indeed, pollution and overpopulation are two major issues for urban areas, which must meet the economic and social challenges of a growing population without degrading the environmental quality in which it evolves. Here again, the IoT is useful:

- home automation;
- digital carriers;
- smart sensors and meters;
- Charging stations for electric vehicles;
- intelligent city lighting, etc.

The applications of connected systems are numerous and it is a real breeding ground for innovation for the IT and digital professions. Within smart cities, the management of individual consumption is better managed, as is the broader treatment of the city's consumption (cost and pollution of public transport and urban travel, waste treatment, electricity management, etc.).

1.5.9 The Internet of Things at the service of road safety.

Particularly popular in recent years, the connected car is making a major contribution to improving road safety. The digital revolution has opened up previously unseen opportunities for the automotive industry.

- Autonomous emergency call box;
- dashboard synchronized with the smartphone;
- application development on dedicated platforms.

Today's car is being transformed into a real computer that gradually leads to the autonomous car, like the one currently being tested at Google. Although our vehicles are not yet capable of driving themselves, they are nevertheless becoming increasingly autonomous thanks to a system that automates certain driving tasks (headlights on, park assist, automatic braking).

1.5.10 Management of its devices with connected accessories.

Connected objects are invading public spaces and businesses, but also our homes, as we have seen. They are also portable and are used in our daily lives for our safety, comfort or simply for our entertainment. This is the case, for example, with Dashbon headsets that can be transformed into masks capable of projecting videos and films being played directly to our eyes on our smartphone. Connected watches and bracelets have also made their way onto the market and are used for many purposes, from entertainment to health, and can even be used as assistance for characters in difficulty.

1.6 IoT Security

Connected devices can give your business a real boost, but anything connected to the Internet can be vulnerable to cyber attacks (X.1361).

1.6.1 Security vulnerabilities and threats in the IoT.

The IoT is the integration of multiple heterogeneous networks, it should address compatibility issues between different networks that are subject to security concerns. The various potential applications of the IoT, the heterogeneity of its enabling technologies and its strong human and socio-economic dimension make its security a difficult and complex subject. Security is an essential element in enabling the widespread adoption of IoT technologies and applications. Without guarantees in terms of confidentiality and authenticity, relevant stakeholders will be unlikely to adopt IoT solutions on a large scale. The IoT is more susceptible to attack than the Internet as billions more devices will produce and consume services. The ubiquity of the IoT will amplify traditional security threats that threat to data and networks, but in addition, the convergence of the physical and virtual worlds through the IoT opens the way to new threats that

will directly affect the integrity of the Objects themselves, infrastructures and processes (physical world), and the privacy of individuals. There are a few categories of threats, which are :

1. Threats to the security of IoT sensors/devices

- **Misuse of the device** :applies to a device that is physically compromised or for which the keys are lost.
- **Sinkhole attack** : means an attack in which a compromised device attracts communication traffic to form a black hole or to implement selective retransmission. In this type of attack, an intruder compromises a device or introduces a counterfeit device into the network and uses it to launch a sinkhole attack. The compromised device attempts to attract all data traffic from neighboring nodes based on the routing parameters used in the routing protocol. Once this is successful, the compromised device will launch an attack. This is a form of network layer attack in which a compromised device sends false routing information to neighboring entities to attract network traffic to it. Because of the presence of ad-hoc networks and the multipoint-to-point communication schemes used by wireless networks where many nodes send data to a single base station, wireless networks are particularly vulnerable to this type of attack. Given the communication flows in a wireless network, the well does not need to target all nodes in the network, but only those close to the base station.
- **Sybil attack** : An attack in which a malicious device illegitimately takes on multiple identities. An additional identity of the malicious device is called a Sybil node. This attack is carried out in conjunction with other attacks to reduce the effectiveness of disruption-tolerant mechanisms such as distributed storage, multi-path routing and topology maintenance.
- **Flooding attacks** : A type of denial of service (DoS) attack in which the attacker sends a succession of "hello" packets to a targeted device in order to consume a sufficient portion of the device's resources so that the device can no longer respond to legitimate traffic.
- **Selective forwarding attack** : An attack in which a compromised node randomly filters received packets and forwards some of them to the next node. A black hole attack occurs when the node filters (discards) all packets it receives.
- **Wormhole attack** : A wormhole attack occurs when two malicious/compromised nodes make it appear that the path between them is very short. A tunnel is a data path between two networked devices that is established within an existing network infrastructure. A network that sends data to another network via a tunnel obtains the data from one network and replicates it to another network through the tunnel and this action can disrupt the network. A hacker can then easily enter the network and use it illegitimately. Combined with a Shaft and Sybil attack, this attack can result in selective retransmission or the creation of a shaft ([Khedim et al., 2018](#)).
- **Sensor/device impersonation** :refers to an attack in which the attacker succeeds in masquerading as a legitimate sensor/device.

2. Threats to the security of IoT gateways

- **Unauthorized access** : Unauthorized access to a gateway may result in the disclosure of sensitive information, modification of data, denial of service and illegal use of resources. For example, once an attacker has successfully gained access to a gateway, he or she can monitor data that is no longer encrypted, compromising usernames, passwords and secure configuration data.
- **Malicious gateway** : Even if all wireless gateways are secure, it is easy for the attacker to deploy his own malicious gateway. For example, an overly “enthusiastic” employee could set up a wireless access point in his or her office without worrying about security, thereby effectively circumventing many of the security measures in place and possibly even causing radio interference to the organization’s and/or company’s official facility. Similarly, a malicious wireless access point could be deliberately and covertly installed to allow the perpetrator of an attack to easily access the network, either locally or remotely. The perpetrator of the attack (known as the “evil twin”) could replace an existing wireless access point with a point that he has full control over the configuration and management of, or even configure, a malicious wireless access point with similar parameters but with a higher power ratio to cover the signal of the legitimate wireless access point. As soon as a legitimate device is fooled and connects to a malicious gateway, it is possible to gather confidential connection information.
- **Denial of service attack** : a denial of service attack saturates the memory and/or computing capacity of its target so that the target significantly slows down or even interrupts the provision of services. The target is busy responding to illegitimate traffic sent to it by the attacker. Wireless sensor networks are particularly vulnerable to denial of service attacks due to their open support characteristics with a dynamically changing topology and the lack of a clear defense strategy. Denial of service attacks are a growing problem for today’s networks. Many of the defense techniques developed for fixed wireline networks cannot be applied in mobile network environments.

3. Threats to network security

- **Unauthorized access** : Unauthorized access to a wireless sensor network may result in the disclosure of sensitive information, modification of data, denial of service and illegal use of resources. For example, once an attacker has gained access to a sensor network, the attacker can monitor data that is no longer encrypted, compromising usernames and passwords.
- **Packet sniffing** : In the case of wireless sensor networks that do not have encryption capabilities, it is usually easy for the attacker to eavesdrop on network communications. This requires an antenna, as well as normal wireless networking tools and a packet sniffer. A packet sniffer is a tool that puts the network card in “promiscuous mode”, which means that the interface will receive and process all traffic, not just the traffic intended for it. The sniffer will show the user all the packets in the network and decode them for easy reading. Plain text traffic is easy to understand and it is possible to set filters to search for certain keywords or values.
- **Bluejacking** : This is an attack on Bluetooth enabled mobile devices, such as cell phones. The attacker sends unsolicited messages to users of Bluetooth devices. The

messages sent do not damage the targeted device, but may cause the user to respond in a certain way or to add new contacts to their phone-book.

- **Bluesnarfing** : This attack allows unauthorized access to information contained in a target wireless device through a Bluetooth connection, often between phones, desktops, laptops and personal digital assistants (PDAs). When successful, this attack can provide unauthorized access to private and confidential information stored on these devices.

4. **Threats to the security of platforms/services** In the Internet, the application layer is primarily responsible for collecting and processing a large volume of user data, including personal information about users or confidential information relating to various transactions (Ado Adamou Abba Ari, 2019). Data is the primary target of the attacker, and can be stolen, altered or damaged. It is necessary to protect it by using confidentiality protection mechanisms. Threats to the application layer include: mass data processing, smart devices that have lost control, unauthorized human intervention, and devices that have lost control and are unable to recover after a disaster. Platform/service-specific threats are :

- **Profiling** : A crawling process used to gather information about the platform/services.
- **Denial of Service** : An attack in which the platform/service is overwhelmed by a very large number of service requests and is therefore too busy to meet the demands of legitimate customers.
- **Arbitrary Code Execution** : An attack in which an attempt is made to execute malicious code on a platform/service in order to compromise the resources of the platform/service and then launch further attacks.
- **Malicious Code Execution** : Any element of a software system or script designed to cause unwanted effects, security or Personal Identity Information breaches, and system damage. Typical examples are viruses, worms and Trojan horses.

1.6.2 Security mechanisms.

The term security encompasses a wide range of different concepts. First, it refers to the basic provision of security services, including confidentiality, authentication, integrity, authorization, non-repudiation and availability. These security services can be implemented through different cryptographic mechanisms such as encryption blocks, hash functions or signature algorithms. For each of these mechanisms, a robust key management infrastructure is fundamental to manage the required cryptographic keys. However, security must focus not only on the security services required, but also on how they are realized in the overall system and how the security features are executed.

1.6.3 State of the art of data security in an Internet of Things and Cloud environment.

The Internet of Things and the Cloud are heterogeneous environments that make it feasible to develop end-to-end solutions to secure data and protect privacy while providing the functionality to collect, store, process and share data (Yahia, 2019). Indeed, on the IoT side, the lack of resources in terms of energy, memory and computing capacity, which characterizes connected objects, limits the deployment of traditional security solutions. Whereas on the Cloud side, resources are relatively unlimited. Nevertheless, the user may be faced with a problem of trust in

the event that the Cloud is managed by a third party. Securing data in an IoT/cloud environment that is heterogeneous is a major challenge that many current researchers are addressing. Several solutions are presented for example for access control in a cloud environment but with the constraints of the Internet of Things, such as the solution proposed by Pussewalage and Oleshchuk (Pussewalage and Oleshchuk, 2016) and Li and al (Li et al., 2010). On the other hand, Gong and al (Gong et al., 2015) and Khemissa and Tandjaoui (et D. Tandjaouig, 2015) propose to secure data and protect privacy in the IoT environment without proposing solutions for data sharing and access control in the Cloud. These examples are a good illustration of the difficulty that the IoT/Cloud environment creates in developing a general and viable security solution. In the solutions cited above, we see that the problem of access control arises at the level of the Cloud, which must be trusted.

1.6.4 Externalization of services.

The externalization mechanism for IoT introduces new challenges in terms of security but also privacy protection. Roman and al (X.1361) list several threats that may impact the security of the IoT ecosystem through outsourcing technology. Some of these threats, such as the possibility of deploying malicious services and the curiosity of “honest” actors, require trust management mechanisms. In summary, in a context of outsourcing of IT services, in addition to traditional (cryptographic) security solutions, the protection of personal data requires a certain visibility on the future behavior of the service provider. This issue can be addressed through trust assessment and service selection systems.

Conclusion

In this chapter, we have spoken briefly about the generality of the Internet of Things. The IoT is the network of things, with clear identification of components, integrated with software intelligence, sensors and ubiquitous connectivity to the Internet. IoT allows things or objects to share information with the manufacturer, operator and/or other connected devices using the telecommunications infrastructure of the Internet. We had gone through some of the security threats that can be posed to connected devices, data or the network itself. However, we will detail some of the protections against these threats in the following chapters.

2. Cryptographic tools used in the IoT

Introduction

The Internet of Things opens up a new world of connected, intelligent devices that can work together to provide virtually unlimited capabilities, and most of these new capabilities will be customizable. Much of the value of the IoT comes from the ability to customize products and services based on a customer's individual and immediate needs. The greatest challenges for IoT will be securing confidential information from unauthorized access and allowing access only to information we are willing to disclose. In order to secure this information, it is necessary to define protection measures to guarantee the confidentiality, integrity and availability of this information. One of the only ways to make the IoT more secure is to use cryptography correctly.

Before getting to the heart of the subject, let's first talk about the generality of cryptography.

2.1 Generality of cryptography

The cryptography uses concepts from many fields (Computer Science, Mathematics, Electronics). However, the techniques are evolving and nowadays regularly find nowadays roots in other branches (Biology, Physics, etc.) (Dumont, 2009; Menezes et al., 1996; Stallings, 2005; tutorialspoint, 2020; wikipedia, 2020).

2.1.1 Vocabularies.

1. **Cryptology** : This is a mathematical science having two branches: cryptography and cryptanalysis.
2. **Cryptography** : Cryptography is the study of methods that make it possible to send data confidentially on a given medium.
3. **Encryption** : Encryption consists in transforming a data (text, message, ...) in order to make it incomprehensible by a person other than the one who created the message and the one who is the recipient. The function for retrieving plain text from encrypted text is called decryption.
4. **Encrypted Text** : Also called cryptogram, ciphertext is the result of applying a encryption to plain text.
5. **Key** : This is the parameter involved and authorizes encryption and/or decryption operations. In the case of a symmetrical algorithm, the key is identical for both operations. In the case of of asymmetric algorithms, it differs for the two operations.
6. **Cryptanalysis** : As opposed to cryptography, its purpose is to retrieve clear text from encrypted texts by determining the flaws in the algorithms used.

7. **Cryptosystem** : It is defined as the set of possible keys (key space), plain text and encrypted possible associated with a given algorithm.

2.1.2 Services of cryptography .

Cryptography has traditionally been used to hide messages from some users. This use is now all the more interesting because communications via the Internet are carried out in infrastructures whose reliability and confidentiality cannot be guaranteed. Cryptography now serves not only to preserve the confidentiality of data but also to guarantee their integrity and authenticity (Menezes et al., 1996).

1. **Confidentiality** : consists in making information intelligible to people other than the actors in the transaction.
2. **Integrity** : verifying the integrity of the data consists in determining whether the data has not been altered during the communication.
3. **Authentication** : consists in ensuring the identity of a user, i.e. to guarantee to each correspondent that his partner is indeed the one he believes to be an access control may allow (for example by means of a password that will have to be encrypted) access to resources only to authorized persons.
4. **Non-repudiation** : information is the guarantee that none of the correspondents will be able to deny the transaction.

2.1.3 Structure of an Encryption Scheme.

In cryptography, the basic property is that $M = D(E(M))$ where :

- M is the plain text, C is the ciphertext;
- K is the key (in the case of a symmetric key algorithm);
- E_k and D_k in the case of asymmetric algorithm ;
- $E(x)$ is the encryption function, and $D(x)$ is the decryption function.

Thus, with a symmetric key algorithm, $M = D(C)$ if $C = E(M)$.

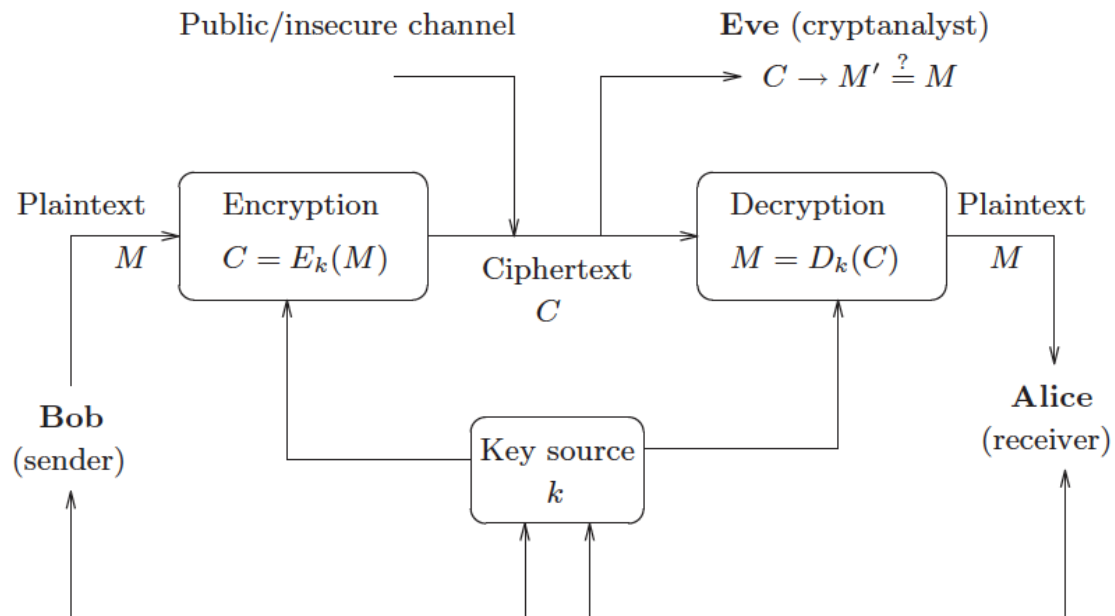


Figure 2.1: Schema of a cryptosystem

2.1.4 Kirchhoff Principle.

The security of the cipher should not depend on what cannot be easily changed. In other words, no secret should reside in the algorithm but rather in the key. Without the key, it must be impossible to retrieve the plain text from the ciphertext. On the other hand, if K is known, decryption is immediate.

This is also known as Shannon's Maxim, derived from the principle stated above: *The adversary knows the system.*

Note : The terms "Secret" and "Robustness" must be distinguished from an algorithm. The secret of the algorithm is to hide the concepts of it, as well as the methods used (mathematical functions). Robustness refers to the algorithm's resistance to various attacks.

2.1.5 Different types of cryptography.

1. Classical Cryptography : Examples of weak encryption algorithms :

- ROT13 (13-character rotation, keyless);
- Caesar's cipher (shift three letters in the alphabet to the left);
- Vigenere's cipher (introduces the notion of key).

Let us give as an example **Cesar's cipher** ([wikipedia, 2020](#)) Caesar encryption is simply an addition in $\mathbb{Z}/26\mathbb{Z}$. Let's set an integer k which is the offset (for example $k = 3$ in the Caesar example above) and define the Caesar cipher function of offset k which goes from the set $\mathbb{Z}/26\mathbb{Z}$ into itself :

$$C_k : \begin{cases} \mathbb{Z}/26\mathbb{Z} & \rightarrow \mathbb{Z}/26\mathbb{Z} \\ x & \rightarrow x + k \end{cases} \quad (2.1.1)$$

To decipher, you just have to go in the reverse direction, i.e. here subtract. Caesar's decoding function of k -offset is:

$$D_k : \begin{cases} \mathbb{Z}/26\mathbb{Z} & \rightarrow \mathbb{Z}/26\mathbb{Z} \\ x & \rightarrow x - k \end{cases} \quad (2.1.2)$$

In fact, if 1 has been converted into 4, by the function C_3 then $D_3(4) = 4 - 3 = 1$. The original number is found. Mathematically, D_k is the reciprocal bijection of C_k , which implies that for all $x \in \mathbb{Z}/26\mathbb{Z}$:

$$D_k(C_k(x)) = x \quad (2.1.3)$$

In other words, if x is a number, we apply the encryption function to obtain the encrypted number $y = C_k(x)$; then the decryption function does what is expected of it $D_k(y) = x$, we get the original number x .

In another way, the ciphered text is obtained by replacing each letter of the original plain text with a letter at a fixed distance, always on the same side, in the order of the alphabet. For the last letters (in the case of a right shift), we start from the beginning. For example, with a shift of 3 to the right, A is replaced by D, B becomes E, and so on until W becomes Z, then X becomes A etc. This is a circular permutation of the alphabet. The length of the offset, 3 in the example mentioned, is the key to the cipher that simply needs to be transmitted to the recipient if he already knows that it is a Caesar cipher so that he can decipher the message. In the case of the Latin alphabet, Caesar's cipher has only 26 possible keys (including the null key, which does not change the text). This is a special case of mono-alphabetic substitution encryption: these substitutions are based on a similar principle, but are obtained by any permutations of the letters of the alphabet. In the general case, the key is given by the permutation, and the number of possible keys is then out of proportion with that of Caesar's ciphers. Caesar's cipher could be used as part of a more complex method, such as Vigenere's cipher. On its own, it offers no security of communication, because of the very low number of keys, which makes it possible to systematically try them when the encryption method is known, but also because, like any mono-alphabetic substitution encoding, it can be very quickly "broken" by frequency analysis (some letters appear much more often than others in a natural language).

Below is an example of a 3-letter rightward encoding. The offset parameter (here 3) is the encryption key :

Letters	J	U	N	I	O	R	K	A	N	I	N	G	I	N	I
Rank plain text(P_k)	9	20	13	8	14	17	10	0	13	8	13	6	8	13	8
Chifered Rank C_k	12	23	16	11	17	20	13	3	16	11	16	9	11	16	11
Ciphered letters	M	X	Q	L	R	U	N	D	Q	L	Q	J	L	Q	L

Table 2.1: Cesar encryption

By using equation 2.1.1, we found :

$$C_3(J) = 9 + 3 = 12 \mod 26 = M$$

$$C_3(K) = 10 + 3 = 13 \mod 26 = N$$

$$C_3(A) = 0 + 3 = 3 \mod 26 = D$$

and by using equation 2.1.2, we found :

$$D_3(M) = 12 - 3 = 9 \mod 26 = J$$

$$D_3(N) = 13 - 3 = 10 \mod 26 = K$$

$$D_3(D) = 3 - 3 = 0 \mod 26 = A$$

2. **Modern cryptography** Today, cryptology is no longer limited to simply encrypting messages to ensure confidentiality: it also focuses on enabling the authentication of various entities communicating remotely in a virtual world, authorizing the signature of intangible digital documents, ensuring data integrity in open networks, protecting the anonymity of sensitive personal data, contributing to the protection of content, and so on. It borrows its methods from various scientific fields, primarily mathematics and computer science, but also physics, especially quantum physics. In this sense, we can speak of the ubiquity of Cryptology.

2.1.6 Symmetric encryption (Private key or secret key encryption) .

Symmetric cryptography uses the same key for the encryption and decryption processes; this key is most often called “secret” (as opposed to “private”) because the entire security of the whole is directly related to the fact that this key is only known by the sender and the recipient. Symmetric cryptography is widely used and is characterized by its great speed (simple operations, encryption on the fly) and by both software and hardware implementations, which clearly accelerates throughput and allows its massive use (Dumont, 2009; Menezes et al., 1996). The general idea of block ciphering is as follows:

1. Replacing characters with binary code;
2. Cut this chain into blocks of a given length;
3. Encrypting a block by “adding” it bit by bit to a key;
4. Move some bits of the block;

5. Repeat operation 3 a certain number of times, if necessary;
6. Move on to the next block and return to point 3 until the entire message is encrypted.

There are three types of encryption per block:

1. **Substitution Encryption** : Substitutions involve replacing symbols or groups of symbols with other symbols or groups of symbols in order to create confusion.
2. **Transposition Encryption** : Transpositions consist of mixing symbols or groups of symbols of a clear message according to predefined rules to create diffusion. These rules are determined by the encryption key. A sequence of transpositions forms a permutation.
3. **Product Encryption** : It's the combination of the two. Encryption by substitution or transposition does not provide a high level of security, but by combining these two transformations a more robust encryption can be achieved. Most symmetric key algorithms use product encryption. A "round" is said to be completed when both transformations have been done once (substitution and transposition).

Advantages : Faster than asymmetric encryption because it uses a smaller key.

Disadvantages : Prior transmission of the key to the correspondent, as many keys as there are correspondents. Security depends on the size of the key. Some symmetrical encryption algorithms widely used:

- Vernam's cipher (the only one offering absolute theoretical security, provided that the key has at least the same length as the message to be encrypted, that it is used only once and that it is totally random);
- DES, 3DES, AES, RC4, RC5, etc...

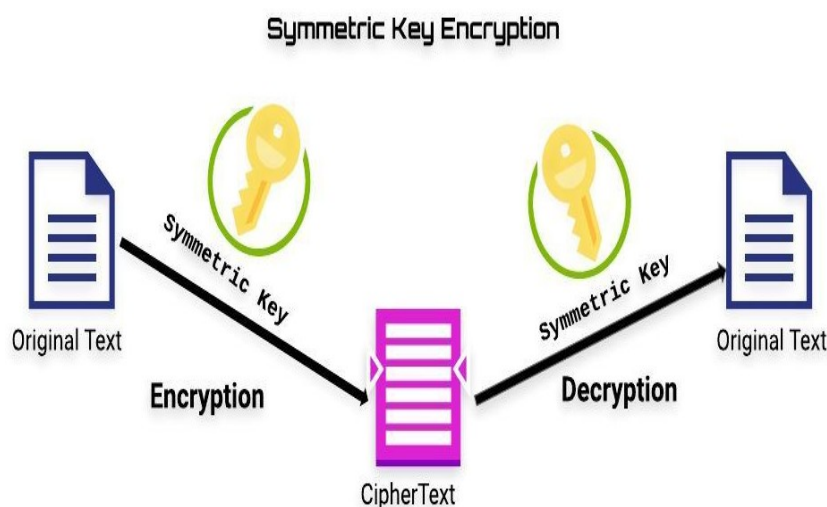


Figure 2.2: Symmetric Key Encryption

- **Data Encryption Standard (DES)**

DES is a symmetrical system that was once a predominant standard in the 1970s, but has since lost favor due to its low security. Its introduction sparked a passionate debate on the role of standards in cryptography and has led to much research and innovation in this field. However, DES is the prototype of block cipher systems, and many systems today are based on its design. DES is a cryptosystem acting in blocks, this means that DES does not encrypt data on the fly when the characters arrive, but it virtually splits plain text into 64-bit blocks that it encodes separately and then concatenates. A 64-bit block of plaintext enters one side of the algorithm and a 64-bit block of ciphertext exits the other side. The algorithm is quite simple since it actually combines only permutations and substitutions. It is a secret key encryption algorithm, the key is therefore used to both encrypt and decrypt the message. This key here is 64 bits long, i.e. 8 characters, but only 56 bits are used. One can therefore possibly imagine a program testing the integrity of the key by using these unused bits as parity check bits. The entire security of the algorithm relies on the keys since the algorithm is perfectly known from all of them. The 64-bit key is used to generate 16 other keys of 48 bits each, which will be used for each of the 16 iterations of the DES. These keys are the same regardless of which block you code in a message. This algorithm is relatively easy to implement in hardware and some chips encrypt up to 1GB of data per second. For industrials, this is an important point, especially when faced with slower, asymmetric algorithms, such as the RSA algorithm, which we will see in detail in the following section. DES has several advantages that have made it the standard symmetric encryption algorithm for a long time, until a few years ago. Here are a few of them: it has a high level of security, it is completely specified and easy to understand, security is independent of the algorithm itself, it is made available to all, by the fact that it is public, it is adaptable to various applications (software and hardware), it is fast and exportable, it relies on a relatively small key, which is used for both encryption and decryption, it's easy to implement.

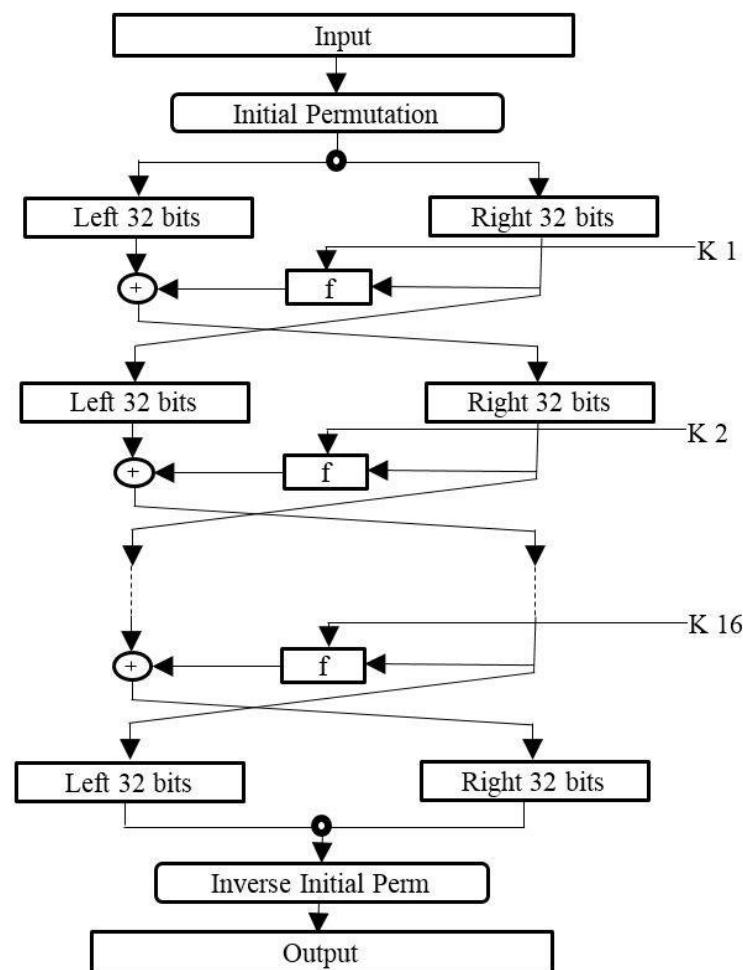


Figure 2.3: Data Encryption Standard (DES) Algorithm (Mushtaq et al., 2017).

The key scheduler : The 56-bit primary key is split into two 28-bit keys. These halves are hereafter treated separately. In each round, each half is rotated left or right by either one or two bits (depending on the round). 24 bits from the left half are chosen, and 24 from the right are chosen to make a 48-bit sub-key. Because we rotate on each round, each bit is only used in approximately 14 out of the 16 rounds. The key scheduler for encryption and decryption are the exact same except that the sub-keys are in reverse order for decryption.

- **Advanced Encryption Standard (AES)** AES is similar to DES in that it is symmetric and uses block ciphers. However, it is much more secure than DES and has become the international standard. It is at least 6 times faster than 3DES. Instead of Fiestel functions, AES uses a substitution-permutation network. This network is a series of operations that either replaces input with output bits (substitution) or shuffles the bits (permutation). It

uses 128-bit input plaintext, but it operates on bytes rather than bits. So, the input is represented as 16 bytes (because 128 bits = 16 bytes), arranged in a 4×4 matrix. This matrix, called the state, will be modified as the algorithm progresses. AES also operates in rounds, but the number of rounds is variable and is based on the length of the key used. A 128-bit key will run AES for 10 rounds, a 192-bit key for 12 rounds, and a 256-bit key will run for 14 rounds. Similar to DES, each round uses a different key. These sub-keys are 128-bits in length and are calculated from the original key. In each round, four transformations are applied:

1. Byte substitution in the status array;
2. Row offset in the status table;
3. moving columns in the status table (except in the last round);
4. addition of a “round key” which varies with each round.

- **Encryption and Decryption**

The sequence of the steps is shown in the following figure

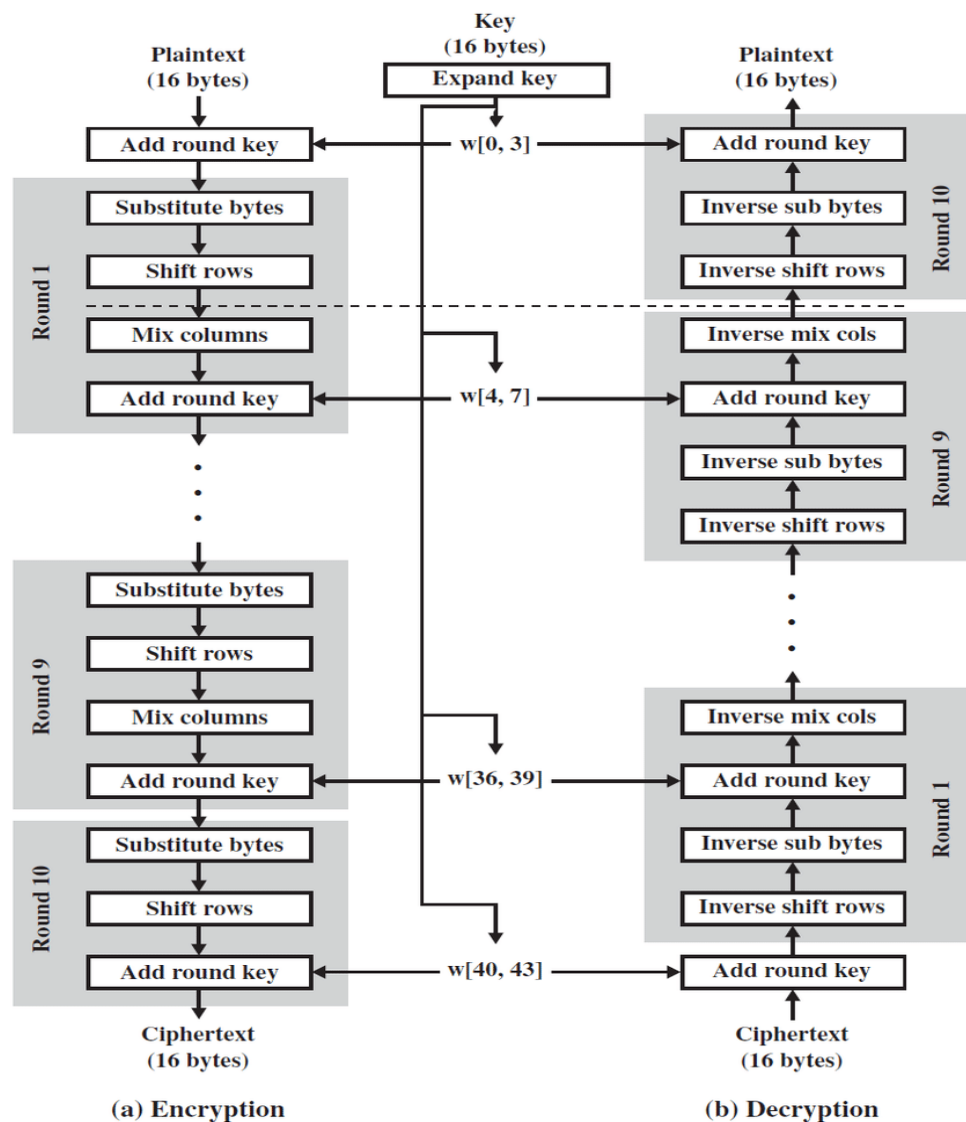


Figure 2.4: Encryption and Decryption process of AES (Mushtaq et al., 2017).

2.1.7 Asymmetric encryption (Public key encryption) .

To solve the problem of key exchange, asymmetric cryptography was developed in the 1970s. It is based on the principle of two keys:

- a public key, allowing encryption;
- a private key for decryption.

As the name implies, the public key is made available to anyone who wishes to encrypt a message.

The latter can only be decrypted with the private key, which must remain confidential. Some widely used asymmetric cryptographic algorithms:

- RSA (encryption and signature);
- DSA (signing);
- ECC, for Elliptic Curve Cryptography (encryption and signature);
- Diffie-Hellman key exchange protocol (key exchange).

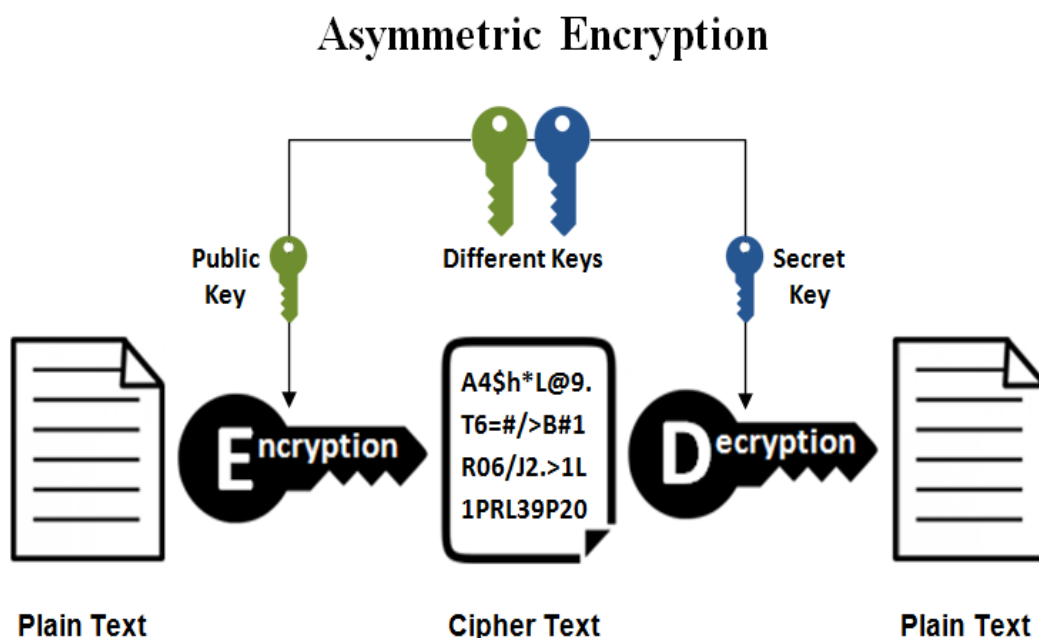


Figure 2.5: Schema of Asymmetric Encryption

- **RSA : Rivest - Shamir - Adleman**

It is based on exponential calculation. Its security is based on the following unidirectional function: the calculation of the product of two prime numbers is easy. The factorization of a number into its two prime factors is much more complex. It is the best known and most widely used system, based on the elevation to a power in a finite field on integers modulo a prime number. The exponentiation number takes about $O((\log n)^3)$ operations which is fast and easy (Menezes et al., 1996). It uses large integers (e.g. represented on 1024 bits). This cryptosystem uses two keys d and e , interchangeable. Encryption is done according to $C = M^e \bmod n$ and decryption by $M = C^d \bmod n$. Safety is based on the cost of factoring large numbers. The number of factorization takes about $O(e^{\log n \log(\log n)})$

operations which requires too much computing time for current machines, in a private setting. It is used for confidentiality, authentication, or a combination of both.

Principles : One has a pair of keys, one public (e, n) and one private (d, n) . The first step is to choose n . It must be a fairly high value, product of 2 very large prime numbers p and q . In practice, if p and q have 100 decimal digits, n will have 200 digits. Depending on the desired security level, the size of n can vary: 512 bits, 768, 1024 or 2048. In a second step, we will choose a very large integer e , relatively prime to $(p - 1) * (q - 1)$. The public key will be formed by (e, n) . Then, we will choose a d such that $e * d \equiv 1 \pmod{\phi(n)}$. The private key will be given by (d, n) . Last phase: throw out p and q . As the cryptanalyst has to find these values, they must be destroyed to avoid leaks.

Summary:

1. Generation of 2 prime numbers p and q ;
2. Calculation of $n = p * q$;
3. Determine e such that $3 < e < \phi(n)$ and $(e, \phi(n)) = 1$;
4. Calculate d such that $e * d \equiv 1 \pmod{\phi(n)}$;
5. Public key: (e, n) ;
6. Private key: (d, n) ;
7. p and q must be kept secret or even deleted ;
8. $C = M^e \pmod{n}$ and $M = C^d \pmod{n}$.

The main disadvantage of RSA and other public key algorithms is that they are very slow compared to secret key algorithms. For example, RSA is 1000 times slower than DES. In practice, for confidentiality purposes, it is used to encrypt a random number which is then used as a secret key for a symmetric encryption algorithm. This is the principle used by software such as PGP(Pretty Good Privacy) for example.

2.1.8 The using of elliptic curves.

It is a concept proposed in 1985 by two researchers, Miller and Koblitz, completely independently. This type of cryptography, still based on the asymmetric model, allows both encryption and signing (Cohen et al., 2005). The abbreviation ECC, for Elliptic Curve Cryptography, is often used. The keys used are shorter for equal or better security. The underlying theory and the implementation are more complex, which explains why this technology is less widespread. However, due to the need to process information more quickly, to manage large amounts of data, and to provide a high level of security, the technology is less widely used and to miniaturize to the maximum, the advantages of this technique are driving research. In general, on R , the elliptic curves will be considered as the set of pairs (x, y) such that :

$$y^2 = x^3 + ax + b$$

whose discriminant $-(4a^3 + 27b^2)$ is not null.

To draw it, for a and b fixed, we compute y such that :

$$y = \sqrt{x^3 + ax + b}$$

1. **Geometric definition (Dumont, 2009)** Let an operation (the addition, +) for the set $E(a, b)$ such that a and b meet the condition of the discriminator. If 3 points on a EC are aligned, their sum is O (point to infinity).

- (a) O is the identity for the addition: $O = -O$.
- (b) For any point P $P + O = P$.
- (c) The opposite of a point $P(x, y)$ is $P(x, -y)$.
- (d) To add two points P and Q , draw a straight line connecting them.

To add two points P and Q , draw a straight line connecting them. This gives us an intersection point R . We define the addition as $P + Q = -R$. Therefore, we define $P + Q$ as the opposite of this point R .

2. The Elliptic Curve on \mathbb{Z}_p

The variables and coefficients take values in the set $[0, p - 1]$ for a certain prime number p , and where all operations are calculated modulo p .

The equation becomes : $y^2 \mod p = (x^3 + ax + b) \mod p$

We note $E_p(a, b)$ the set of pairs of integers (x, y) that satisfy this equation. This is called an **elliptic group**.

Example : Let $p = 11$ and the elliptic curve $y^2 = x^3 - x + 4$. As we are working in \mathbb{Z}_p , the pairs (x, y) satisfying the equation are the following :

$E = (0, 2), (0, 9), (1, 2), (1, 9), (4, 3), (4, 8), (5, 5), (5, 6), (6, 4), (6, 7), (9, 3), (9, 8), (10, 2), (10, 9) \cup P_\infty$ For all points $P, Q \in E_p(a, b)$:

- (a) $P + O = P$
- (b) If $P = (x_P, y_P)$, then $P + (x_P, -y_P) = O$ and $(x_P, -y_P) = -P$.
- (c) $P = (x_P, y_P)$, and $Q = (x_Q, y_Q)$ with $P \neq -Q$, then $R = P + Q = (x_R, y_R)$ is determined as follows:

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P} \right) \mod p & \text{if } P \neq Q \\ \left(\frac{3x_P^2 + a}{2y_P} \right) \mod p & \text{if } P = Q \end{cases} \quad (2.1.4)$$

- (d) The multiplication is defined as a repetition of additions (e.g. $3P = P + P + P$).

Note : The previous section discusses the problem in \mathbb{Z}_p . The equations are different if we work in the finished field $GF(2^m)$.

3. Elliptic Curve Cryptography (ECC)

To use elliptic curves in cryptography, one must find a difficult problem (such as factoring a product into its prime factors in the case of RSA) (Cohen et al., 2005; Dumont, 2009). Consider the equation $Q = kP$ where $Q, P \in E_p(a, b)$ and $k < p$. It is easy to calculate Q knowing k and P , but it is difficult to determine k if we know Q and P . This is the **problem of the discrete logarithm for elliptic curves** : $\log_p(Q)$. In a real use, the k is very large, making the brute force attack unusable.

- (a) **ECC for key exchange** Let k be a large prime integer q (considering that we will use the equations presented previously, and not the equations in $GF(2^m)$) and the parameters a and b satisfying the equation $y^2 \bmod q = (x^3 + ax + b) \bmod q$. This allows us to define $E_q(a, b)$. Let us then take a starting point $G(x_1, y_1)$ in $E_q(a, b)$ whose order n is high. The order n of a point on a EC is the smallest positive integer such that $nG = O$. $E_q(a, b)$ and G are made public. The exchange of a key per ECC between two entities A and B is as follows:
- A chooses an n_A less than n which will be its private key. A then generates its public key $P_A = n_A * G$.
 - B chooses an n_B less than n to be its private key. B then generates its public key $P_B = n_B * G$.
 - A generates the secret key $K = n_A * P_B$ and B generates the secret key $K = n_B * P_A$.
- (b) **ECC for encrypting data** Although elliptic curve cryptography is often used to exchange a symmetric key, it is also used to directly encrypt data (Cohen et al., 2005). Here is an example of a cryptosystem using them. Here, the plain text m must be encoded as a point P_m of x and y coordinates. It is this point that will be encrypted. Here too, a point G and an elliptical group $E_q(a, b)$ must be made public. Users must also choose a private key and generate the corresponding public key. To encrypt the message, A randomly determines a positive integer k and produces C_m as a pair of points such that $C_m = kG, P_m + kP_B$. Notice the use of B 's public key. To decipher, B must multiply the first point by his private key, and subtract the result from the second point received : $P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$.

2.1.9 Hash functions.

One of the fundamental primitives in modern cryptography is the cryptographic hash function, often informally called a one-way hash function (Menezes et al., 1996). A simplified definition for the present discussion follows.

Definition : A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values.

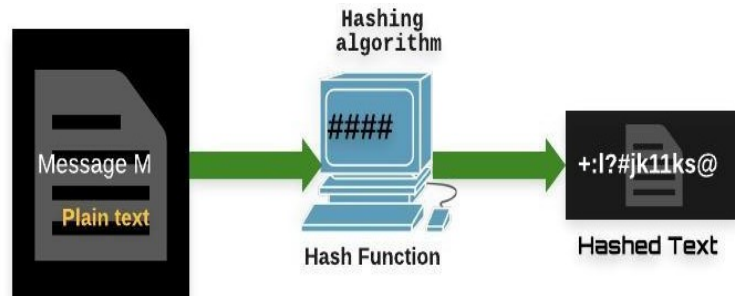


Figure 2.6: Hash function

1. **Features of Hash Functions** The typical features of hash functions are ([tutorialspoint, 2020](#)):

- **Fixed Length Output (Hash Value) :**
 - Hash function converts data of arbitrary length to a fixed length. This process is often referred to as hashing the data.
 - In general, the hash is much smaller than the input data, hence hash functions are sometimes called compression functions.
 - Since a hash is a smaller representation of a larger data, it is also referred to as a digest.
 - Hash function with n bit output is referred to as an n -bit hash function. Popular hash functions generate values between 160 and 512 bits.
- **Efficiency of Operation**
 - Generally for any hash function h with input x , computation of $h(x)$ is a fast operation.
 - Computationally hash functions are much faster than a symmetric encryption.

2. **Properties of Hash Functions** In order to be an effective cryptographic tool, the hash function is desired to possess following properties :

- **Pre-Image Resistance :** This property means that it should be computationally hard to reverse a hash function.
In other words, if a hash function h produced a hash value z , then it should be a difficult process to find any input value x that hashes to z .
This property protects against an attacker who only has a hash value and is trying to find the input.
- **Second Pre-Image Resistance :** This property means given an input and its hash, it should be hard to find a different input with the same hash.
In other words, if a hash function h for an input x produces hash value $h(x)$, then it should be difficult to find any other input value y such that $h(y) = h(x)$.
This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.
- **Collision Resistance :** This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
In other words, for a hash function h , it is hard to find any two different inputs x and y such that $h(x) = h(y)$.
Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
This property makes it very difficult for an attacker to find two input values with the same hash. Also, if a hash function is collision-resistant then it is second pre-image resistant.

The main applications of hash function are : security of passwords, integrity, authentication process, implementation of randomized function in cryptosystem, digital signature, post-quantum cryptography. We can conclude this part by listing some of the usual hash functions : MD5, SHA1, SHA2, SHA3 .

2.1.10 Digital Signatures.

1. **Definition** Message authentication protects both communicating entities from an intruder. However, it does not protect them from each other. In the event of a conflict, nothing prevents the repudiation of the message(s). Two examples illustrate this principle:

- Entity A could forge a given message and say that it comes from B. All it has to do is create a message and apply a MAC to it using the key they have in common.
- The opposite case may also exist. B may deny having sent a message, for the simple reason that the construction explained in the previous point is possible. It is therefore impossible to prove that such a message was or was not sent by B rather than by A.

This is where digital signatures come in. In particular, they make it possible to verify the author, the date and time of the signature, to authenticate the contents of a message and can be verified by third parties to resolve conflicts.

2. **Properties** A digital signature must

- depend on the signed message;
- use unique sender-specific information to prevent forgery and denial;
- be relatively easy to produce, recognize and verify;
- be mathematically unfeasible to forge (by constructing new messages for an existing digital signature, or by constructing a fraudulent digital signature for a given message);
- be easy to store.

2.2 Cryptography, data protection in the IoT environment

2.2.1 Cryptographic techniques used for IoTs.

Cryptography is probably the most used technique in traditional wireline and wireless networks with a considerable computing and memory capacity. Cryptography solutions are known as safe solutions that provide answers to all data security problems. The specificities of sensor networks, i.e. low computing power and limited memory in addition to the problem of energy conservation, are considerable obstacles to the use of common cryptographic systems known to be safe (SSL, RSA, etc.).

Symmetric key encryption solutions are exploitable within sensor networks and provide a real solution for network security. However, while symmetric key encryption is possible within sensor networks, the total security of this type of solution remains to be demonstrated. On the one hand because symmetric key encryption does not guarantee data authentication as the digital

signature of public key encryption can do, and on the other hand because the problem of the distribution of encryption keys arises. Thus, while the IEEE 802.15.4 protocol specifies the encryption method to be used, at no time does it specify how the keys should be managed and how to enable data authentication (Ado Adamou Abba Ari, 2019; Dumont, 2009). Asymmetric cryptography mechanisms are potentially good security solutions when the number of nodes is very high, such as sensor networks. They are less problematic for key management. However, asymmetric cryptography useful for establishing shared keys between nodes is not realistic. The demands on computing time and memory are so high in this case in addition to the need for a centralized infrastructure for key management.

In the IoT environment, trusting any object is a challenge. Security can be implemented at both the software and hardware levels. Most of the security issues in IoT are the same and familiar as those in the current Internet, but their impact may be different. Implementing a security solution in software is much easier and easier to maintain than a hardware solution. The implementation of cryptographic techniques is possible at both the hardware and software level. Cryptographic techniques are fundamentally easier to implement in software, but software implementations are easier to temper because they work on the top layer of the operating system (OS). Instead of traditional cryptographic techniques in the IoT environment, Lightweight Cryptography is used for smart objects.

2.2.2 Lightweight Cryptography.

Some researchers have introduced a new variety of identity-based encryption scheme (IBE), called fuzzy IBE (Sahai and Waters, 2005). It allows a private key to identify the ciphertext to be decrypted and also to encrypt the identity if the two are identical and much closer together, it is measured by the metric “overlapping sets”. This scheme uses the biometric input as a private key that is subject to noise. The security of the blur-based IBE system is proven by a selective identification template. The error tolerance between the public and private key is used for the encryption process.

The Attribute Based Encryption (ABE) system allows the user’s private key to be expressed in any access formula above attribute (Ostrovsky et al., 2007). The proof of security is based on Diffie-Hellman’s bilinearity hypothesis. The performance of this system and the previous system can be compared and it is less resource intensive.

However, ABE is an expensive encryption method in terms of computing capacity. Indeed, while it does not pose any particular problems for use in the Cloud, this is not the case for an implementation on resource-constrained devices such as connected devices. For example, attribute-based encryption (ABE) schemes are generally computationally intensive and are not suitable for resource-constrained devices.

Several recent researches that have been done in this problematic that these different types of encryption pose, the results have shown that the use of public key cryptography based on elliptic curves (ECC) are better placed to solve this problem (Hammi et al., 2020; Hankerson et al., 2006). This is because the problem of the discrete logarithm on a well-chosen elliptic curve is faster than that of the discrete logarithm in finite fields. The algorithm known to be the most efficient to solve such a problem is exponential time, unlike the RSA system for which there are sub-exponential time algorithms. An algorithm is sub-exponential if the logarithm of the execution

time grows asymptotically slower than any given polynomial. The security level of a key is in direct correspondence with its size (Faye, 2014). Thus, for the same resistance, elliptic curves require shorter keys than RSA. However, the use of smaller keys confers many advantages: calculations are faster, overall power consumption is reduced, and memory space is reduced. Unlike RSA, elliptic curves are faster for private operations than for public operations. RSA requires less time for encryption and digital signature verification than decryption and signature generation, which are slower and require more resources. On the contrary, elliptic curves in cryptography require more computation during encryption and signature verification than decryption and signature generation. Nevertheless, not only encryption and verification are effective, but also decryption and signature generation. Thus, they are practical in resource-constrained environments such as sensor networks, smart cards, mobile phones, etc. ECCs are expected to progressively replace RSA (Vaidya et al., 2009). The following table presents a comparison in terms of key size of ECC and RSA cryptosystems for the same level of security.

RSA keys(bits)	ECC keys(bits)
1024	160
2048	224
3072	256
7680	384
15360	521

Table 2.2: Comparison between ECC and RSA (NIST parameters)

2.2.3 IoT Standards and Protocols.

The Internet of Things covers a wide range of business sectors and use cases, from a single constrained device to massive multi-platform deployments of integrated technologies and cloud-based systems connecting in real time. Many communication protocols, old and new, that enable devices and servers to communicate with each other in new and more interconnected ways, link all of these elements.

1. Protocols

We give a general overview of some of the common protocols of the IoT architecture and we will give some of the security protocols used in the IoT.

- **Infrastructure** ex: 6LowPAN, IPv4/IPv6, RPL)
- **Identification** (ex: EPC, uCode, IPv6, URIs);
- **Transport** (ex: Wifi, Bluetooth, LPWAN);
- **Discovery** (ex: Physical Web, mDNS, DNS-SD);
- **Data Protocols** (ex: MQTT, CoAP, AMQP, Websocket, Node)
- **Device Management** (ex: TR-069, OMA-DM)
- **Semantic** (ex: JSON-LD, Web Thing Model)
- **Multi-layer Frameworks** (ex: Alljoyn, IoTivity, Weave, Homekit)

2. Protocols for IoT Security

There are some essential protocols to secure IoT. These protocols are applicable to execute both symmetric as well as asymmetric cryptographic algorithms.

- **HTTP** : Hyper Text Transfer Protocol is a keystone of client server-based protocol for the Internet, which is used only in the client side of IoT to initiate connection but not to receive connection request.
- **XMPP** : Extensible Messaging and Presence Protocol defines a novel concept in IoT. It is especially applicable for voice and video calls as well as strengthening in the sense of addressing security and scalability of IoT systems.
- **CoAP** : It is applicable to define miserable nodes and is also capable to manipulate existing recourse with the help of interfaces. Some functions of TCP are also replicated in it, although it uses user datagram protocol.
- **MQTT** : Message Queue Telemetry Transport has been designed and developed for unreliable network system having low bandwidth. It controls the overall network from backend server.
- **AMQP** : Advance Message Queuing Protocol is applicable for queuing and routing in IoT-based system for point to point communication and security.

2.3 Gap in research

In the literature review, we examined the problems for data protection and privacy security within an IoT network and some proposed solutions. Finding an adequate solution to this problem requires a balance between the security measures that can be supported and the question related to the different constraints we find to the connected devices, for example whether there are enough resources left for the functionality of the devices (e.g. a small sensor). A lot of work has been done to secure IoT devices, but given the constraints of connected devices, users may not have full assurance for privacy security or data protection within the IoT.

Conclusion

Cryptography opens up new areas of possibilities for networking with devices with fewer resources. IoT is a new era technology and the concern for security will always be present whenever there is a proposal for a new technology. Although not all aspects of security are new, changing platforms and connectivity, power consumption, use of limited resources, the strength of trust over connected objects with limited control over authorization and authentication are changing the sphere of protection and methods of protection.

The Internet of Things promises to link everything together, everywhere and everywhere. All devices must interact effectively with each other in a secure, scalable and reliable manner. In fact, with today's centralized architecture, it could be difficult and challenging to manage the scalability of huge IoT networks. This problem can be solved by adopting a new approach to

security, different from the current centralized model. As the objective of this work is to propose “smart, lightweight cryptographic schemes to enhance trust and security in the Internet of Things”, we will focus on one of the emerging new security solutions for the Internet of Things that will allow us to ensure privacy security and guarantee authentication, data integrity, confidentiality and non-repudiation of data in the IoT. This new emerging approach addresses extensibility, interoperability and compatibility issues very effectively: Blockchain technology. The solution we will propose will be based around the use of the blockchain in the IoT to ensure privacy security.

3. Research Methodology

Introduction

The centralized architecture poses problems in case the central processing unit fails. The network will no longer be accessible once the central processing unit or central node fails. Adequate solutions should be found to overcome this problem. This thesis will offer a solution for the security of the IoT network infrastructure, communication security and privacy security and also the challenges faced by IoT devices with many constraints to ensure security in the broadest sense. The proposed solution will be based on a decentralized architecture using Blockchain technology which will be able to guarantee privacy security, data protection and also the security of the IoT network infrastructure. The proposed solution will take the form of a method involving asymmetric encryption and a security gateway that acts as an intermediary between the devices and the Internet. This solution will ensure the confidentiality, integrity and authenticity of the data transported over the Internet and the confidentiality of the data as it passes between the device and the gateway. The objective of this research is to propose intelligent and lightweight cryptographic systems to enhance trust and security in the Internet of Things, so the results of this research will serve as a guide for the development of similar systems. We hope that this thesis will be useful and informative for those who will read it and that it may stimulate further research in this field. This thesis will also contribute to the existing pool of knowledge in the field of IoT security.

3.1 Blockchain for IoT Security and Privacy

3.1.1 Background.

Many virtual currencies and cryptocurrencies use the blockchain for their security. Satoshi Nakamoto, the inventor of Bitcoin, was the first to apply a decentralized blockchain. A blockchain is a technology for the storage and transmission of information without a control device. Technically, it is a distributed database in which information sent by users and internal links within the database are checked and grouped at regular time intervals into blocks, thus forming a chain. The whole is secured by cryptography. The blockchain was first introduced in 2008 as the technical basis for a cryptographic currency known as Bitcoin, and since then it has been widely used in other cryptocurrencies.

By extension, a blockchain is a distributed database that manages a list of records protected against tampering or modification by the storage nodes; it is therefore a secure, distributed record of all transactions carried out since the start of the distributed system. In other words, the blockchain is a technology that provides perennial, tamper-proof, distributed digital data storage.

As a basic technique for all kinds of encryption currency, the blockchain is built on a distributed digital transaction ledger that is held by all participating entities in a peer-to-peer network. Decentralization is implemented by deploying all participating entities to verify and confirm new transactions. Once verified, confirmed and recorded, transaction data cannot be retroactively

changed without altering all subsequent blocks, which requires consensus of the majority of the network. The batches of valid transactions are formed in blocks. Each block includes the hash results of the previous block that connects the two adjacent blocks. A hash function is nothing more than a mathematical function that allows from a digital file (text, drawing, images, video, etc.) to obtain a string called a 256-bit hash.

The linked blocks then form a chain called blockchain. The process of building a blockchain consists of continuously adding new blocks to the existing blockchain, which is also called mining. The basic operation of mining is to solve a mathematical problem (usually a proof of work (PoW)) that is difficult to solve but easy to verify. In order to solve this problem, the participating entities have to provide enormous computational resources that limit the number of blocks that can be mined.

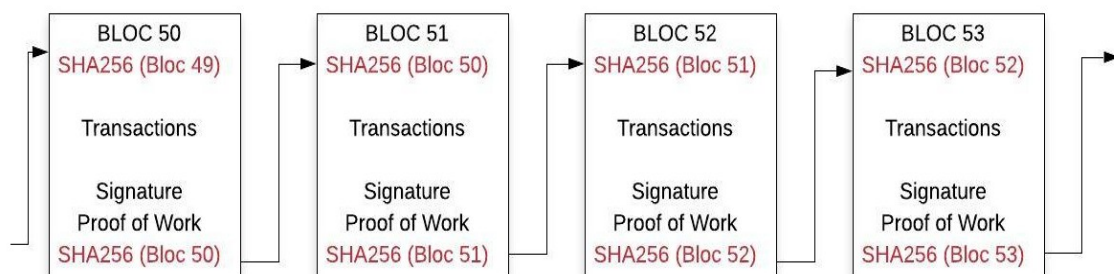


Figure 3.1: Example of the linked blocks

This mechanism also prevents malicious extraction of blocks. The popular puzzle used in blockchains is usually Proof of Stake (PoS). In order to solve this problem, the participating entities have to provide enormous computational resources that limit the number of blocks that can be mined. POW requires significant computational resources that are deployed in the Bitcoin protocol to find the specific value with specific hash results. PoS will consume both computing and memory resources. All messages exchanged between the entities are encrypted by deploying modifiable public keys to prevent eavesdropping.

Since blockchains are a technology that can deploy many applications using many famous encryption currencies such as Bitcoin, there are other potential applications that can deploy blockchain as a core technology. As the blockchain can enable payment without trusted intermediaries, many financial services such as digital assets, money transfers and smart contracts are being developed. In this work, the blockchain will be applied to the design of a communication and interaction system in the Internet of Things (IoT) to ensure privacy security.

The blockchains offer the following advantages in IoT security:

- **Decentralization** : Due to the decentralized architecture of the IoT, the blockchain is the most suitable security solution for the IoT. The decentralized architecture of the blockchain

makes the security solutions more scalable and can solve the single point of failure problem and becomes more robust to denial of service attacks.

- **Pseudonymity** : Nodes in the blockchain are identified by their public keys (or public key hash). These pseudonyms do not provide information about the identity of the participating nodes.
- **Transaction Security** : Each transaction, before being sent to the Blockchain network, is signed by the node and must be verified and validated by minors. After validation, it is virtually impossible to falsify or modify transactions already recorded in the blockchain. This makes it possible to prove the traceability of events in the system.(Sahai and Waters, 2005)

3.1.2 Architecture.

Before presenting the general architecture combining IoT and the blockchain, this part will explain in a general way the architecture of IoT and the blockchain. Several architectures have been proposed by different researchers. There is no single consensus on the IoT architecture, which is universally agreed upon.

- **Three-layer and five-layer IoT architectures**

The most basic architecture is a three-layer architecture which was introduced in the early stages of research in this field. It consists of three layers, namely the perception, network and application layers.

1. **The perception layer** : is the physical layer, which includes sensors to detect and collect information about the environment. It detects certain physical parameters or identifies other intelligent objects in the environment.
2. **The network layer** : is responsible for connecting to other smart objects, network devices and servers. Its characteristics are also used for the transmission and processing of sensor data.
3. **The application layer** : is responsible for the provision of user-specific services. It defines various applications in which the Internet of Things can be deployed, such as smart homes, smart cities and smart health.

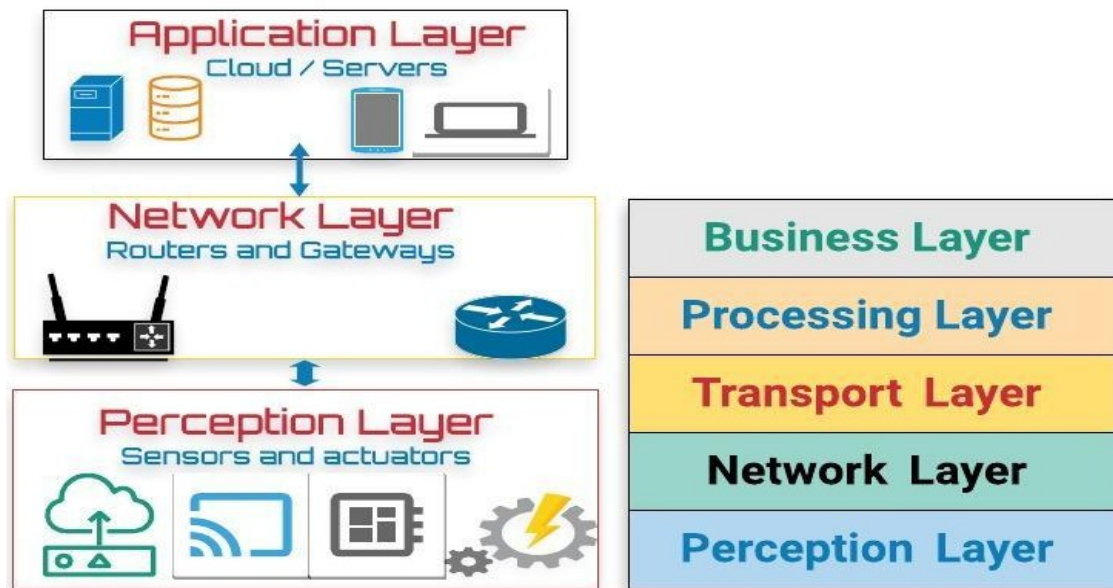


Figure 3.2: Three-layer and Five-layer architecture of IoT

With the rapid development of the IoT, many auxiliaries and business models have emerged. The rudimentary three-layer architecture model cannot provide a sufficiently accurate abstraction. Accordingly, a more concrete five-layer architecture is proposed to fulfill the description of IoT. The five layers are perception, transport, processing, application, and business layers. The role of the perception and application layers is the same as the architecture with three layers. We outline the function of the remaining three layers.

1. **The transport layer** : transfers the sensor data from the perception layer to the processing layer and vice versa through networks such as wireless, 3G, 4G, 5G LAN, Bluetooth, RFID, and NFC.
2. **The processing layer** : is also known as the middleware layer. It stores, analyzes, and processes huge amounts of data that comes from the transport layer. It can manage and provide a diverse set of services to the lower layers. It employs many technologies such as databases, cloud computing, and big data processing modules.
3. **The business layer** : manages the whole IoT system, including applications, business and profit models, and users' privacy. The business layer is out of the scope of this paper. Hence, we do not discuss it further manages the whole IoT system, including applications, business and profit models, and users' privacy.

- **Blockchain Architecture**

We propose a five-layer architecture model. This model is illustrated in Figure 3.3.

The physical layer is composed of various types of blockchain nodes. They interconnect with each other to form the blockchain network. In general, we usually have two types of nodes in a blockchain: full nodes and light nodes.

1. **Full nodes** : keep a copy of the complete blockchain in their memory. They participate in creating, checking and adding new blocks to the blockchain. As a result, full nodes maintain security, consensus, and data accuracy in the blockchain. Network participants require a high level of physical storage capacity to become a complete node. Some complete nodes may be miners who contribute their computing power and resources to achieve a relative amount of reward. The miner performs a hash calculation and transmits its result to the blockchain network. Then the result will be verified by others under a consensus protocol. Once the result is confirmed, the miner receives his reward.
2. **Light nodes** : Light nodes do not require high hardware specifications. They do not keep a copy of the complete blockchain, but simply know the status of the last block. The light nodes get the blockchain information from the complete nodes.

The network layer provides a reliable data delivery service. It allows the nodes in the blockchain to communicate with each other. In general, the blockchain network is based on a peer-to-peer network. There are no central servers or central authority in the network. Network participants enter into an agreement to ensure the integrity and security of the network through a consensus protocol. As a result, the blockchain allows decentralization and avoids centralized vulnerabilities.

The consensus layer is responsible for ensuring the consistency of the stored data and provides an incentive for participants to find a new block. The consensus mechanism is the important part of the blockchain to ensure the accuracy and integrity of the data.

The propagation layer consists of communication protocols that define the rules for the propagation of messages and blocks in the network.

Bitcoin and Hyperledger fabric adopt Gossip as their communication protocol while Ethereum uses the Kademlia algorithm as the basis of its communication protocol.

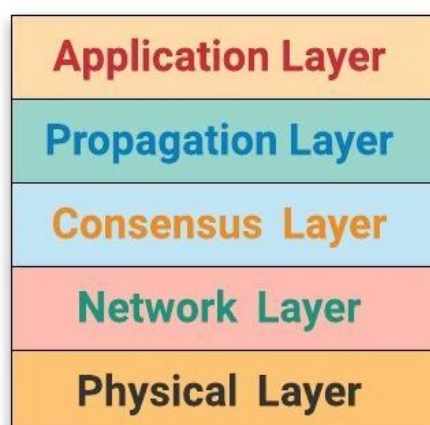


Figure 3.3: Five-layer architecture of Blockchain

- **Gossip Protocol** : is a procedure or process of computer peer-to-peer communication that is based on the way epidemics spread. Some distributed systems use peer-to-peer gossip to ensure that data is disseminated to all members of a group. Some ad-hoc networks have no central registry and the only way to spread common data is to rely on each member to pass it along to their neighbors. The term epidemic protocol is sometimes used as a synonym for a gossip protocol, as gossip spreads information in a manner similar to the spread of a virus in a biological community.
- **Kademlia** : Kademlia is an algorithm designed for decentralized peer-to-peer networks. It is used in Ethereum to optimize network routing and help locate target nodes. In Kademlia, each node has its own unique identifier. The Kademlia algorithm uses the Node ID to perform a quick peer-to-peer search and locate files or resources. In the Kademlia network, each node is recognized as a leaf on a binary tree. The logical distance between two Node IDs can be calculated by applying the XOR operation. The smallest result represents the closest logical distance between two nodes. Nodes with the longest common prefix (LCP) will share a sub-tree. K-buckets can be expressed as sub-trees. The K in K-bucket refers to the maximum number of leaves in the sub-tree. The routing table in Kademlia stores the information in a K-bucket list.

3.1.3 Consensus.

The term “consensus” means that all nodes in the network must agree on an identical version of the blockchain. Somewhere, the consensus mechanism of a blockchain is an internal and automatic audit of its network. This protocol is therefore essential and has two functions:

- It allows the blockchain to be updated while ensuring that every block in the chain is valid. The people involved in validating the blocks (called the “nodes” of the network) must have an incentive to get involved in the security of the network, with this incentive generally taking the form of a monetary reward.
- It prevents one and the same entity from controlling the entire network and thus guarantees its decentralization.

1. The rules of a consensus protocol

A block consensus follows specific rules. Network nodes will have to ensure that each block and each transaction meets a number of criteria before they can be validated. Each piece of data must receive the unanimous acceptance of all network nodes before it can be validated. The main objective of consensus protocols is to be able to fight against attacks that threaten the security of the network. A very common example of an attack is double-spending. It consists of issuing two transactions that spend the same asset. The role of the consensus is then to decide which of the two transactions can be considered as validated and which should, conversely, be cancelled.

2. The main mechanisms of blockchain consensus

The consensus of a blockchain is defined from its creation by its founders. The most famous and first of them is the Proof-of-Work (PoW), which is used in the Bitcoin protocol.

- **The Proof-of-Work (PoW)**

The Proof-of-Work protocol is the most widely used of all blockchain consensus. Since 2009, it has been able to demonstrate its resistance and security against various attack attempts. In the Proof-of-Work protocol, the individual nodes of the network are referred to as miners. In order to confirm a transaction, the miners must solve a complex mathematical problem that requires a lot of computing power. To do this, they use a mathematical procedure called a hash function. The hash function is used to write the transaction data into blocks and connect them together. There are different types of hash functions, such as the SHA 256, used on Bitcoin. Once the hash is written into the blockchain, it is forgery-proof. A miner is rewarded for each block he manages to approve and confirm. His income is proportional to the computing power he is able to deploy to solve the problem.

- **The Proof-of-Stake (PoS), or proof of issue**

Proof-of-Stake, or proof of stakes, is based on a completely different logic than proof of work and does not require any particular computing power. In Proof-of-Stake, the participants of the consensus can be assimilated to shareholders of a business entity having the privilege of participating in its consensus mechanism. PoS is a much less expensive consensus than PoW, as it does not require any special energy or hardware costs. Concretely, to validate a block, the nodes must prove their possession of a certain amount of cryptography and pledge it on the network. The more this quantity is important, the more likely a node will be chosen to update the register of a blockchain. The PoS consensus is that these people are the most likely to want to fight a network attack that could ruin them entirely.

- **Delegated Proof of Stake (DPoS)**

Emerging more recently, the Delegated Proof-of-Stake tends to address the weaknesses of PoS and PoW by offering a hybrid model. The DPoS consensus operates on the same basic principle as PoS. However, those responsible for forging blocks must be elected by community members. The election system ensures that the blockchain is not controlled by a minority of people, as can be the case of a miner with a large amount of computing power, or a PoS forger with a very large amount of tokens. The DPoS protocol has many advantages, but also some disadvantages.

Numerous consensus such as Proof of Work, Proof of Stake and Proof of Delegated Stake are the main consensus in the blockchain. There are many others such as **Byzantine fault tolerance (BFT)** which is the characteristic of a distributed network that allows consensus (agreement on the same value) to be reached even when some of the nodes in the network do not respond or respond with incorrect information. The objective of a BFT mechanism is to guard against system failures by using collective decision making (both correct and defective nodes) that aims to reduce the influence of the defective nodes.

3.2 Proposed architecture

The architecture we propose in Figure 3.4 provides the privacy and data protection security mechanism. This architecture also provides a secure mechanism to store data in a decentralized file system using blockchain technology and exchange data between the base station (LPWAN) which is a low-power WAN technology and users wishing to have the data on the basis of Smart Contracts. The data is received by the data logger from many sensors. The data logger sends the encrypted data to the local acquisition node which is computationally powerful. The local acquisition node is integrated into an LPWAN gateway. The peers in this network send data through the gateway, the data is stored in the IPFS which will compute data hashes that will in turn send them to the nodes in the blockchain for storage. The IPFS will store the data and the nodes in the blockchain will store the data hashes.

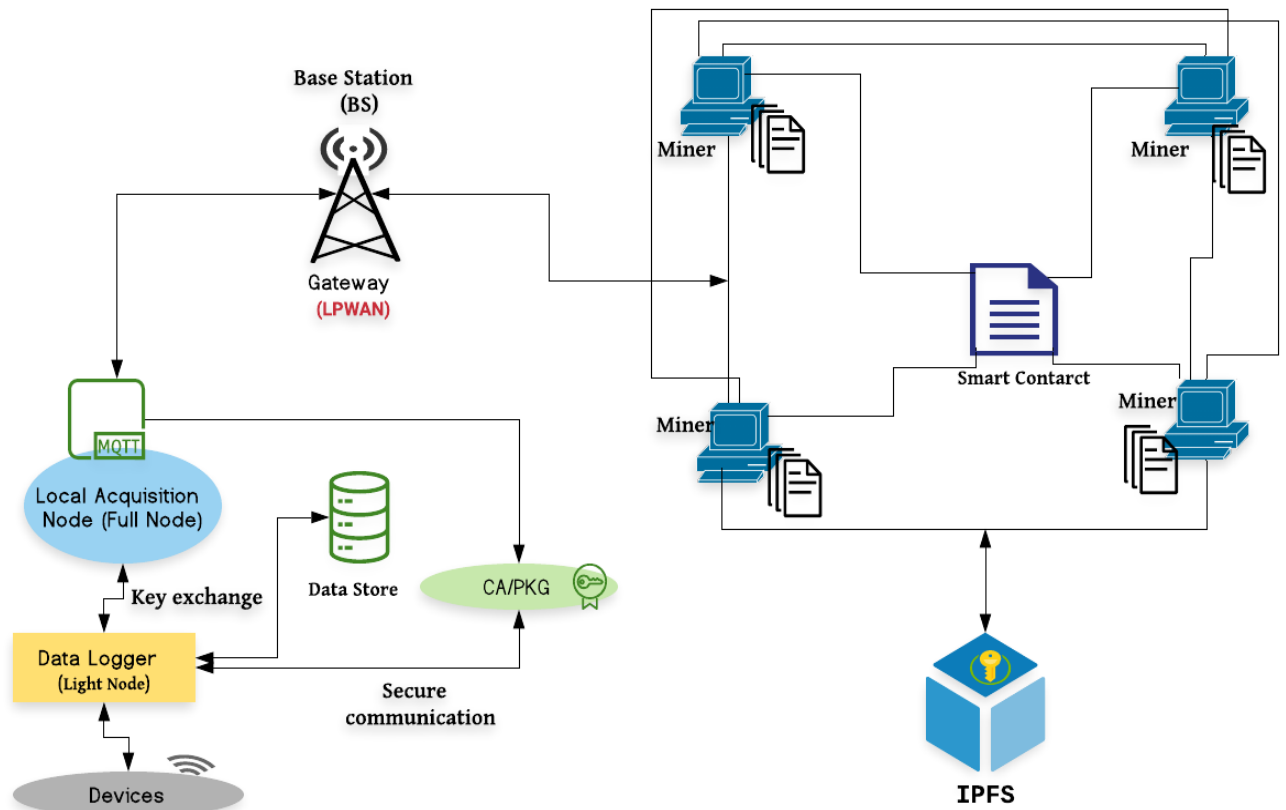


Figure 3.4: Proposed architecture

3.2.1 Sensors (End Devices).

Given the storage constraints, the energy and computing power that the sensors have, it is not

possible to implement blockchain functions in the sensors because they only have detection capabilities. The data received by the IoT /LPWAN gateway is transmitted to a chain infrastructure as shown in Figure 3.4. This is a way to communicate with a blockchain node without any storage or computation requirements in the IoT system. If the Logger node, which is a lightweight node, wants to connect directly into the blockchain network, it will not download the complete blockchain because it is a node that is not regular in the network due to various constraints mentioned above. Instead, it will download the block headers only to validate the authenticity of the transactions.

3.2.2 Local Acquisition Node.

Local Acquisition Node is a full node with computing power and also storage capacity. The primary responsibility of the local acquisition is to acquire the data sent by the data logger. The data received from the data logger is encrypted to ensure confidentiality. Key distribution is performed with the help of the certification authority or key generators (CA/PKG). The local acquisition node sends the transaction through the MQTT protocol to LPWAN, which is a complete node for solving the puzzle or the proof of the work.

3.2.3 Certification Authority (CA).

The CA is responsible for the issuance and management of certificates. It generates the certificates and ensures the integrity and authenticity of the information they contain by identifying itself with its private key. To be able to issue certificates, it must first receive certification requests containing the public key of the requesting Entity.

Wireless sensor nodes are limited by resource constraints; therefore, it is not possible to perform encryption on sensor nodes that result in additional power consumption of thin devices.

3.2.4 LPWAN Gateway (Base Station).

Low Power Wide-Area Networks (LPWANs) are designed to connect devices that require large scale transmission and long battery life with minimal power consumption. There are currently three main networks in operation : LoRaWAN, Sigfox and Cellular LTE Cat M1/NB-IoT. In our case, the gateway is a powerful computing device that acts as a complete node in the blockchain. The main responsibility is to route data to the network and check the integrity of the network to achieve an IoT infrastructure without trust.

3.2.5 Local Storage (Data store).

Local storage is a storage device, such as a backup disk, which is used by devices to store data locally. This storage is associated by the data logger (which is a lightweight node, e.g. raspberry PI). Storage uses a FIFO (First-in-First-out) method to store data and stores the data from each device as a register chained to the device's starting point. This does not prevent a device (sensor, etc.) to directly interact with the blockchain network.



Figure 3.5: Raspberry associated with an external storage disk([framboiseaupotager.blogspot, 2020](https://framboiseaupotager.blogspot.com/2020))

3.2.6 InterPlanetary File System(IPFS).

IPFS is a distributed P2P¹ network that provides a high-capacity content-addressable block storage model using hyperlinks for access, forming a generalized acyclic Merkle oriented graph². IPFS combines a hash table, encouraged block exchange and a self-certified namespace. It stores data and provides a 32-byte hash. In order to preserve the confidentiality of the required data, the hash is then encrypted with the private key of the gateway, which is considered here as a full node, and transferred to the blockchain. Each node of the IPFS stores the data it is interested in. Figure 3.6 shows the different colors of the nodes indicating their interest in the type of data. The data stored in the IPFS can only be accessed by its hash, which is only available to authenticated users.

¹<https://www.supinfo.com/articles/single/2787-ipfs-web-p2p>

²<https://github.com/ipfs/ipfs/blob/master/README.md>

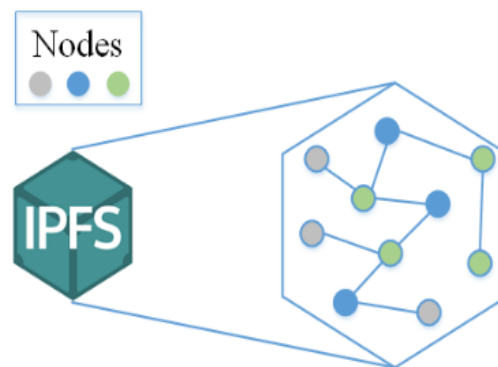


Figure 3.6: IPFS's Structure

IPFS is a combination of several technologies such as Distributed Hash Tables, a Block Exchange system, a Version Control system.

Distributed Hash Table (DHT) is a distributed system. The principle is to provide a “Hash Table”, i.e. a set of “key-value” pairs in a distributed environment, so that each node of the blockchain can retrieve any value associated with a key. The association of a key with a value being distributed among the nodes of the set makes it possible to form a system that does not require any particular coordination between the nodes.

For the Block Exchange, the mechanics of BitTorrent will have inspired IPFS. BitTorrent’s data exchange protocol promotes sharing: in the figure 3.4, a node that shares a lot of data will also have priority to receive it. The system therefore invites nodes to be active in the network.

These two mechanisms together form a very participative ecosystem, and therefore very interesting to distribute the bandwidth of users.

The Version Control tools allow you to trace the evolution of a file over time. The use of a “Merkle Tree” (which could be summarized very simply as a “Hash Tree”) ensures the validity of the transaction (information in the IPFS).

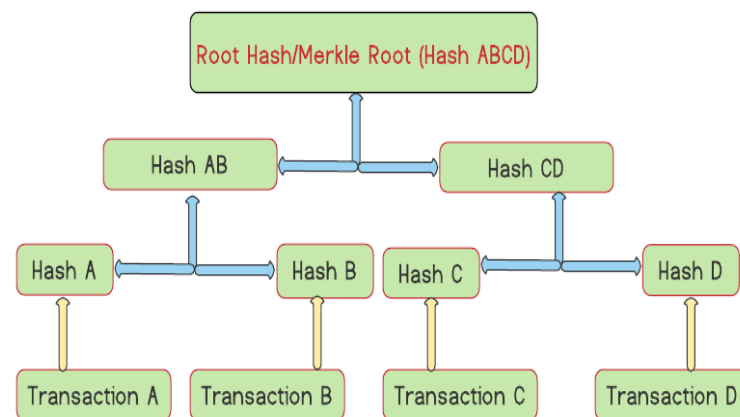


Figure 3.7: Blockchain Merkle Tree Root

3.2.7 The layered architecture derived from the proposed architecture.

The architecture presented in the Figure 3.3 gives in detail the 5 layers of the blockchain we considered. Figure 3.8 shows a modular blockchain-IoT architecture in which each layer is decoupled from the other layers so that developers can replace or add any new module without affecting the rest of the system. You will notice that the propagation and consensus layers we presented in the five-layer architecture are combined to give the IoT-blockchain layer shown in Figure 3.8. The IoT physical layer consists of various linked devices with communication, computing and data storage capabilities. The main function provided by the connectivity layer is routing management, as self-organization is necessary because the physical devices themselves do not have global Internet protocols. This layer also contains other modules for providing services, including network management, security management and message broker. The IoT-blockchain service layer contains all the modules that organize common services to provide various functionalities of blockchain technologies, including identity management, consensus, and peer-to-peer (P2P) communication. The distributed registry is a consensus of replicated, shared, and synchronized digital data that propagates across the blockchain network, where all network participants can have their own copy of the registry. It also provides a secure storage space to record device configuration and detection data from the physical sensors. All changes made to the ledger are reflected in all copies in minutes or, in some cases, seconds. The general ledger can be authorized or unauthorized, to determine who is authorized to execute a peer to validate transactions (Hang and Kim, 2019). The IPFS module allows the blockchain to be an efficient mode for storing large data. The intelligent contract is a kind of code invoked by an external client application to manage access and changes in the general ledger. It is typically installed and instantiated on each counterpart in the network. Event Management sends events each time a new block is added to the General Ledger or triggered each time the condition predefined in the Smart Contract is met. The API interface exposes the services provided by the blockchain network as services through which the client application can access and manage the network. The top layer is the application layer, where various interfaces are provided to view data from physical devices, to manipulate

and control the devices.

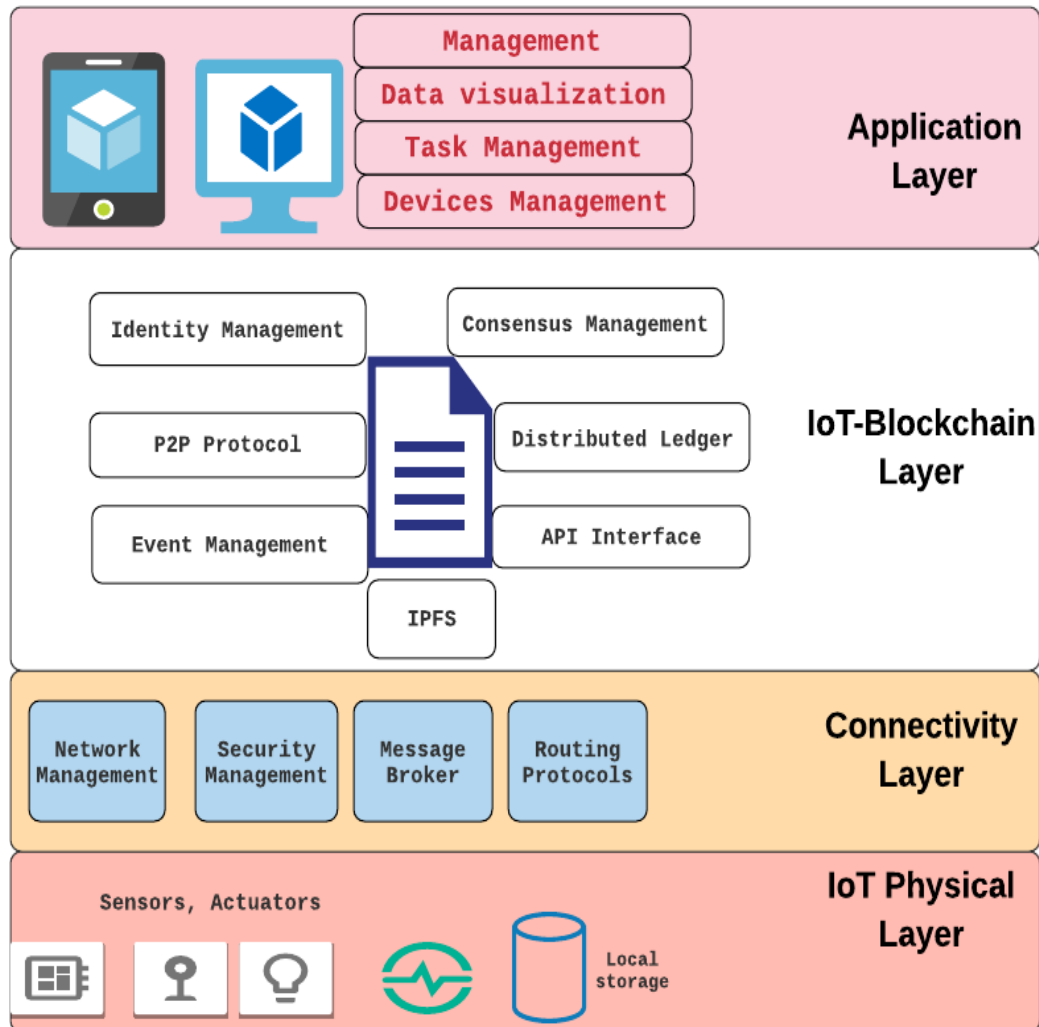


Figure 3.8: Architecture of the Blockchain-IoT platform based on layers

3.3 Elliptic curve-based cryptographic protocol for secure communication and privacy security

As we can see in our architecture presented in Figure 3.4, we notice that security is reinforced in each communication layer. Hence we need to define a mathematical model to guarantee the exchange of cryptographic keys for the nodes (objects) participating in the network and the encryption of information.

The use of mathematical tools, such as elliptic curves, for cryptographic purposes only makes sense if they are embedded in secure protocols. Asymmetric cryptography computations are less efficient compared to symmetric cryptography. Thus, we generally use asymmetric cryptography to establish a secure channel between two entities, *obj1* and *obj2*, which will then use symmetric cryptography to exchange messages. This is due to the various constraints we encounter in the connected devices, including low computing power, imitated energy and also low storage. A channel is considered secure if there is mutual authentication or not between *obj1* and *obj2*, and construction of session keys that will be used by the symmetric cryptography to exchange messages.

3.3.1 Key exchange protocols.

It is an exchange of keys in the manner of Diffie and Hallman shown in Figure 3.9, i.e. without communicating them directly to each other. *ObjA* and *ObjB* agree together and publicly on an elliptic curve $E(a, b, K)$, i.e. they choose a finite field \mathbb{K} (e.g. $\mathbb{Z}/p\mathbb{Z}$) and an elliptic curve $y^2 = x^3 + ax^2 + b$.

They also choose together and always publicly a point P located on the curve.

Then, each on its own and secretly, *ObjA* chooses an integer a and *ObjB* chooses an integer b . *ObjA* sends to *ObjB* the point of the elliptic curve aP and *ObjB* sends to *ObjA* the point bP . Each on its side is able to compute $a(b)P = b(a)P = (ab)P$. This point is their secret key. Their public keys are respectively aP and bP .

The DH protocol is then called ECDH for Elliptic Curve Diffie-Hellman where the secret shared between *objA* and *objB* is the coordinates of the abP point.

Note that this protocol is sensitive to the so-called Man-In The-Middle (MITN) attacks which consist in usurping the *ObjA* or *ObjB* identity by intercepting aP or bP which are sent in clear text. To protect against this kind of attack, it is important to authenticate the sender of aP or bP using a signature protocol.

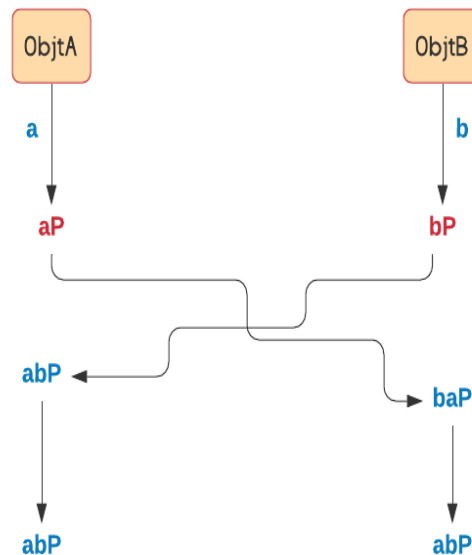


Figure 3.9: Diffie-Hellman key exchange protocol

3.3.2 Signing Protocols.

In order to authenticate an entity and guarantee the integrity of an information or guarantee the non-repudiation of an information, it is required to use a signature protocol. A signature protocol is divided into two parts:

1. **Signature** : The first part consists of the construction of a signature from a private key and the message to be signed. This operation is sensitive because the calculations performed manipulate secret data such as the private key of the signing entity. If this key leaks, an attacker will then be able to forge signatures in the place of the legitimate entity.
2. **Verification** : the second is to verify that a signature s associated with a message m is valid, i.e. validating that s is mathematically related to m and the public key associated with the private key that was used to sign m . This operation does not manipulate only public data, so the verification of a signature does not will not be susceptible to physical attacks because the private key is not used.

Given an elliptic curve E and a point P of order n on it, we can construct the private/public key pair composed of a random number a for its private part and point aP for its public part. This key pair is of the same shape for all the signature protocols on the elliptic curves that we will consider in the rest of this study.

There are several signature protocols using elliptic curves such as the Elliptic Curve Digital Signature Algorithm (ECDSA) or the Edwards Digital Signature Algorithm (EdDSA). These protocols are standardized by NIST for ECDSA or by an RFC in the case of EdDSA.

A signature constructed from this protocol is a pair (R, s) and is generated from :

- E: the parameters defining the elliptic curve used.
- P: a point of the curve of high order.
- n: the order of the point P
- hash: a hash function.

A signature of a message m is then computed using a private key a :

$$\begin{aligned} R &= rP \\ s &= (r + \text{hash}(R, m)a)[n] \end{aligned}$$

The verification of the signature (R, s) of message m is done using the key public Q associated with private key a .

Thus, we need to compute :

$$R' = sP - \text{hash}(R, m)Q$$

and verify that :

$$\text{hash}(R', m) = \text{hash}(R, m)$$

This protocol is correct because:

$$\begin{aligned} R' &= sP - \text{hash}(R, m)Q \\ &= sP - \text{hash}(R, m)aP \\ &= (r + \text{hash}(R, m)a)P - \text{hash}(R, m)aP \\ &= rP \\ &= R \end{aligned}$$

1. Algorithm ECDSA signature

Input : Private key a , Message m

Outputs: (r, s) the signature associated with m

$$\begin{aligned} H(m) &= \text{hash}(m) \\ \text{Do} & \\ & k = \text{random between } 1 \text{ and } n - 1 \\ & (x, y) = kP \\ \text{While } x &= 0 \\ & r = x[n] \\ & s = k^{-1}(H(m) + ar)[n] \\ \text{While } s &= 0 \\ \text{Return } & (r, s) \end{aligned} \tag{3.3.1}$$

2. ECDSA verification

Input : m a message, (r, s) the associated signature, Q the public key associated with the private key that was used to sign the message.

Outputs: Returns true if the signature is correct, false if not.

$$\begin{aligned}
 H(m) &= \text{hash}(m) \\
 u &= H(m)s^{-1}[n] \\
 v &= rs^{-1}[n] \\
 (x, y) &= uP + vQ \\
 \text{If } x &= r \text{ then} \\
 &\quad \text{Return False} \\
 \text{End If} \\
 &\quad \text{Return True}
 \end{aligned} \tag{3.3.2}$$

We can show that a signature produced by the algorithm 3.3.1 will be well validated by algorithm 3.3.2

$$\begin{aligned}
 uP + vQ &= H(m)s^{-1}G + rs^{-1}Q \\
 &= H(m)s^{-1}G + rs^{-1}aG \\
 &= (H(m) + ar)s^{-1}G \\
 &= (H(m) + ar)k(H(m) + ar)^{-1}G \\
 &= kG \\
 &= (x, y)
 \end{aligned}$$

3.3.3 Encryption and Decryption.

Taking into account the various constraints we mentioned at the beginning of this section, encryption and decryption will be done using symmetric cryptography because of its performance and also the low power consumption and also the size of its key which is reduced. In this work, we made the choice on the AES encryption algorithm. The key exchange will be done using asymmetric cryptography with elliptic curves.

4. Ethereum Blockchain

In this thesis, we made the choice of the Ethereum blockchain which was created in 2014 by Vitalik Buterin. It is the second most capitalized cryptography after Bitcoin, which also works on the open source model, with a decentralized network. Ethereum is mainly used for the creation of smart contracts and decentralized applications. It is also at the origin of the creation of tokens called ERC-20 which allow other projects to launch their ICOs (Initial Coin Offering).

The table below shows some differences between Bitcoin and Ethereum.

Characteristic	Bitcoin	Ethereum
Year Introduced	2009	2015
Inventor	Satoshi Nakamoto	Vitalik Buterin, Gavin Wood
Currency Unit	Bitcoin	Ether
Smallest Unit	Satoshi	Wei
Symbol	btc	eth
Total supply	21 Millions btc	Unlimited
Block creation time	10 minutes	10 - 12 Seconds
Intended use	Payment Network	Programming platform (Dapp)
Mining Reward	12.5 btc	3 Eth
Mining Algorithm	SHA-256	Ethash

Table 4.1: Differences between Bitcoin and Ethereum

In the Ethereum Blockchain, instead of exploiting Bitcoin, the miners focus on acquiring Ether, a type of encryption token that powers the network. Beyond a tradable crypto-currency, the Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.

There is a second type of token used to pay miners' fees for including transactions in their block, which is gas, and each smart contract execution requires the sending of a certain amount of gas to induce the miners to set it up on the Blockchain.

Smart contracts, a central element of Ethereum, are stand-alone programs that, once started, automatically execute predefined conditions. They function like any "if - then" conditional instruction. The benefits: increased speed, improved efficiency, and confidence that the contract will be executed as agreed. These programs are able to overcome moral hazard problems, and reduce the costs of verification, enforcement, arbitration, and fraud.

The advantage of implementing smart contracts in a blockchain is the guarantee that the terms of the contract cannot be changed. A smart contract that is not in the blockchain would be a program whose terms could be changed during execution.

4.1 Mining Algorithm in Ethereum

Ethereum uses the Eth-Hash algorithm¹ to find cryptographic hash for a block (extraction process). Before the start of the extraction process, a large acyclic directed graph (DAG) is created. The mining process tries to solve a certain condition of it. This process is a proof of work (PoW) in the ether and is designed to be checked by other nodes very quickly in linear time on a CPU using fewer resources.

Nonce's miners guess that Nonce builds a block and adds it to the chain. It is incredibly difficult to guess the exact nuncio, so a difficulty value is set and changed after each block. If a block is extracted before the average extraction time (10 to 12 seconds), the difficulty increases and decreases when the extraction time exceeds the average extraction time.

4.2 Genesis Block in Ethereum

The genesis block is the first or original block of a private network on Ethereum. The genesis block contains all the essential information to configure the network as well as to find peers linked to it. It is essentially the configuration file of the proposed Ethereum network. In fact, to start your network, you must actually indicate the location of the file as a parameter.

The genesis file is a simple JSON file that contains configuration thresholds. Here is an example :

```
{
  "config" : {
    "chainId" : 15,
    "homesteadBlock" : 0,
    "difficulty" : "1",
    "gasLimit" : "9999999",
    "nonce" : "0xdeadbeefdeadbeef",
    "mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
    "alloc" : {
      "7df9a875a174b3bc565e6424a0050ebc1b2d1d82" : {
        "balance" : "30000"
      },
      "f41c74c9ae680c1aa78f42e5647a62f353b7bdde" : {
        "balance" : "40000"
      }
    }
  }
}
```

- **config** : The file begins with the "config" block which contains all the configuration parameters and thresholds that control the basic operations of the network.
 - **chainId** : Protects the network from a replay attack. It acts as an offset to prevent attackers from deciphering continuous values in the network.

¹<http://gavwood.com/paper.pdf>

- **homesteadBlock** :Homestead is the second major version of Ethereum (the first version is Frontier).
- **difficulty** : This determines the difficulty of extraction in the network
- **gasLimit** : This sets the limit of gas cost per block.
- **nonce & mixhash** : Nonce and mixhash are values which, when combined, allow to verify that a block has really been cryptographically exploited and therefore that it is valid. The mixhash is a 256-bit hash that proves, when combined with the 64-bit nonce, that a sufficient amount of computation has been performed on this block: the proof of work (PoW).
- **alloc** : Allows to define a list of pre-filled wallets. This is a specific feature of Ethereum to manage the “ether pre-sale” period. We can also leave this list as an empty hash.

There are many other parameters and configuration options that can be added to the genesis block.

4.3 Ethereum Network

The basis for decentralized consensus is the peer-to-peer network of participating nodes that maintain and secure the blockchain ([ethdocs](#), 2020).

4.3.1 Statistics of the Ethereum network.

EthStats.net is a dashboard of live statistics from the Ethereum network. This dashboard displays important information such as the current block, hash difficulty, gas price and gas expenses. The nodes displayed on the page are only a selection of real nodes on the network.

4.3.2 Ethereum network types.

Nowadays, most Ethereum projects rely on Ethereum as a public blockchain, providing access to a wider community of users, network nodes, currencies and markets. However, there are often reasons to prefer a private blockchain or a consortium blockchain.

1. **Public Blockchain** A public blockchain is a blockchain that anyone in the world can read, anyone in the world can send transactions and expect to see them included if they are valid, and anyone in the world can participate in the consensus process, the process of determining which blocks are added to the chain and what the current status is. As an alternative to centralized or quasi-centralized trust, public blockchains are secured by cryptoeconomics (thus referring to the new form of economy also called “economy 2. 0” brought about by crypto-money through Blockchain technology, and initiated by the first Bitcoin blockchain), the combination of economic incentives and cryptographic verification using mechanisms such as proof of work or proof of participation, according to a general principle that the degree to which someone can have influence in the consensus process is proportional to the amount of economic resources they can contribute.

2. **Consortium Blockchain** A consortium blockchain is a blockchain where the consensus process is controlled by a set of pre-selected nodes; for example, one could imagine a consortium of 15 financial institutions, each of which operates one node and 10 of which must sign each block for the block to be valid. The right to read the blockchain can be public or limited to participants, and there are also hybrid routes such as root hashes of blocks being public with an API that allows members of the public to make a limited number of queries and retrieve cryptographic evidence of parts of the blockchain state. These blockchains can be considered “partially decentralized”.
3. **Private Blockchain** A fully private blockchain is a blockchain where write permissions are centralized within an organization. Read permissions can be public or arbitrarily limited. Likely applications include database management, auditing, etc. internal to a single company, so public readability may not be necessary in many cases, although in other cases public auditability is desired.

Even though these private/consortium blockchains may not have a connection with the public blockchain, they still contribute to the overall Ethereum ecosystem by investing in the development of Ethereum software.

4.3.3 Geth.

Geth is a client of Ethereum, set up in Golang and used to connect to the Ethereum network. Other C++ and Python implementations are also available. Once installed, it can be launched with the genesis block (the first block of the block chain), booted and connected to the different networks available in Ethereum. These networks are identified by their network ID. Different nodes having the same genesis block and the same network ID can synchronize with each other and perform transactions. Network ID: 1 is for the main network, 2 and 3 are reserved for test networks. A different network ID can be used to set up a private network with a different Different versions for Windows, Linux, MacOS and Android are available.

4.3.4 Connecting to the network ([ethdocs](#), 2020).

In a network, Geth continually tries to connect to other nodes in the network until he has peers. If UPnP is enabled on the router or by running Ethereum on a server connected to the Internet, it will also allow connection from other nodes. Geth discovers peers through something called the discovery protocol. To find other nodes in the geth network, use protocol discovery. In the discovery protocol, nodes communicate with each other to discover other nodes in the network. Initially, geth uses a set of bootstrap nodes whose ends are stored in the source code. We can use the Administration API to perform node-related operations such as getting node information, listing all peers, adding a new peer to the network.

Verification of Connectivity and ENODE IDs

To check the number of peers the client is connected to in the interactive console, the net module has two attributes that give you information about the number of peers and tell you whether you are a listening node:

```
> net.listening : which sends a Boolean value
> net.peerCount : sends the number of peers
```

To check the node information like to check the ports used by geth, we use the geth console to execute the command “admin.nodeInfo” as shown below.

```
{
> admin.nodeInfo
Name: 'Geth/v0.9.14/darwin/go1.4.2'
NodeUrl: 'enode://3414c01c19aa75a34f2dbd2f8d0898dc79d6b219ad77f8155abf1a287ce2ba60f14998a3a98c0cf14915eabfdacf914a92b27a01769de18fa2d049dbf4c17694@[::]:30303',
NodeID: 3414c01c19aa75a34f2dbd2f8d0898dc79d6b219ad77f8155abf1a287ce2ba60f14998a3a98c0cf14915eabfdacf914a92b27a01769de18fa2d049dbf4c17694',
IP: '::',
DiscPort: 30303,
TCPPort: 30303,
Td: '2044952618444',
ListenAddr: '[:]:30303'
}
```

To obtain more information about the connected peers, such as IP address and port number, supported protocols, use the peers() function of the admin object.

```
[{
> admin.peers
ID: 'a4de274d3a159e10c2c9a68c326511236381b84c9ec52e72ad732eb0b2b1a2277938f78593cdbe734e6002bf23114d434a085d260514ab336d4acdc312db671b',
Name: 'Geth/v0.9.14/linux/go1.4.2',
Caps: 'eth/60',
RemoteAddress: '5.9.150.40:30301',
LocalAddress: '192.168.0.28:39219'
}, {
ID: 'a979fb575495b8d6db44f750317d0f4622bf4c2aa3365d6af7c284339968eef29b69ad0dce72a4d8db5ebb4968de0e3bec910127f134779fbcb0cb6d3331163c',
Name: 'Geth/v0.9.15/linux/go1.4.2',
Caps: 'eth/60',
RemoteAddress: '52.16.188.185:30303',
LocalAddress: '192.168.0.28:50995'
}]
```

4.3.5 Smart contract in layer-based Blockchain-IoT architecture.

The smart contract was first introduced by Nick Szabo in 1994, which is defined as 'a computerized transaction protocol that executes the terms of a contract (Cha et al., 2018). In the context of the blockchain, the smart contract acts as a trusted distributed application that gains its trust from the blockchain and the underlying consensus among peers. Since they reside on the blockchain, smart contracts have a unique address through which the end user can send a transaction to it. Depending on the data that triggers the predefined condition, the smart contract then executes automatically and independently in the manner prescribed by each peer in the network.

In principle, smart contracts are written in non-standard or domain-specific language (such as Solidity) in order to achieve consensus among all peers. This becomes one of the biggest challenges to the widespread use of the smart contract, as blockchain developers have to learn a new language to write smart contracts, which can lead to various coding problems. In addition, the performance and scale of transaction execution is limited because all transactions are executed successively by all peers. To solve these problems, we deploy smart contracts on a specific subset of peers rather than on all peers, so the transaction only needs to be executed by a set of peers. This approach also supports parallel execution, which can significantly increase overall performance and system scale. In addition, we use standard languages such as Node.js or Java to code the smart contract so that developers can use their familiar programming languages without spending time learning a new language.

The proposed smart contract contains various functions that allow users to interact with the general ledger, which is a combination of the state database and the blockchain. For example, users can create, update, and query information about the device from the general ledger by submitting transactions to the smart contract. It also provides functionality to manage the transactions offered by the devices, such as collecting detection data or updating actuator status changes. A block contains a hash value of the transactions and the hash value of the previous block to ensure the security of the ledger data. Even if the ledger hosted by a peer is tampered with, it could not convince all other peers because the ledger is distributed over a network. An example of a registry structure is shown in Figure 4.1, where the blockchain contains four blocks. Block 0 is the genesis block, which contains no transactions. Each of the other blocks contains one transaction, and these transactions are associated with various assets (e.g., sensor, actuator) in the state of the ledger. The application running on the smart contract receives the transaction and performs various types of queries and updates. The transaction is added to the block and, in the meantime, the general ledger report is updated. In the end, the result of the general ledger update is returned to the application as a response.

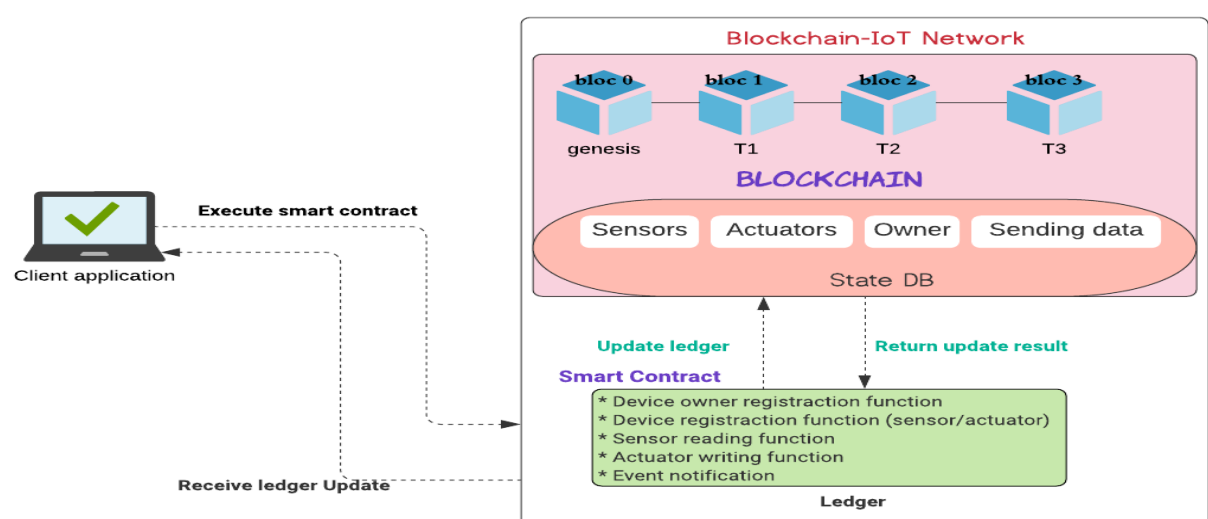


Figure 4.1: smart contract

4.4 Sequence diagram of the different operations in the proposed architecture

In this section, we present the different scenarios of information flow in our system. The scenarios are presented in several ways: the first is that a device (sensor/actuator) sends information (data) to the network, i.e. the information to be recorded in the blockchain by the InterPlanetary File System (IPFS). Second, we present the scenario in which a user wishes to access the blockchain network. Thirdly, a user queries the blockchain network to store the IPFS data and lastly, we present the different steps to access the data stored in the blockchain network by the InterPlanetary File System (IPFS).

Before transmitting data, the device first authenticates itself to the authentication entity before connecting to the base station, once done, the identity integrity check by the certification authority is performed before the device accesses the IoT-Blockchain network. The data is stored in the IPFS and the IPFS returns the hashed data which is sent to the smart contract to trigger the consensus process. Once the transaction is validated by the miners, the transaction will be distributed and recorded in the blockchain ledgers. Figure 4.2 illustrates these different scenarios.

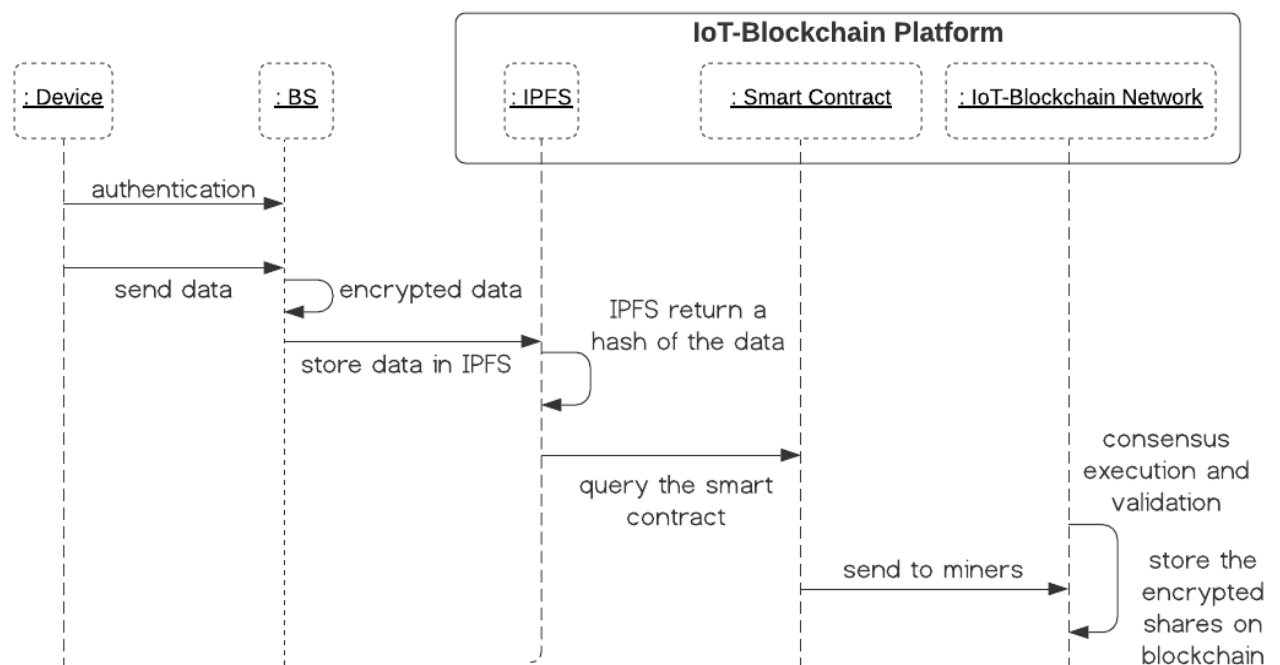


Figure 4.2: Sequence diagram of the different operations

Before a user carries out a transaction in the blockchain, the user must have proof of identity. Figure 4.3 presents the processes for registering the identity and registering the owner of the device. To obtain the identity, the user submits the registration request to the blockchain network. This request is processed by the identity management module, which issues a secret for the registration process via the client application. The registration request is then sent by the client, using the identification ID and secret obtained during the registration procedure. The identity management service transmits the certificate of enrollment (ECert) with the public key as a response. The ECert is used to request the transaction certificate (TxCert), and finally the TxCert is returned to sign the transactions.

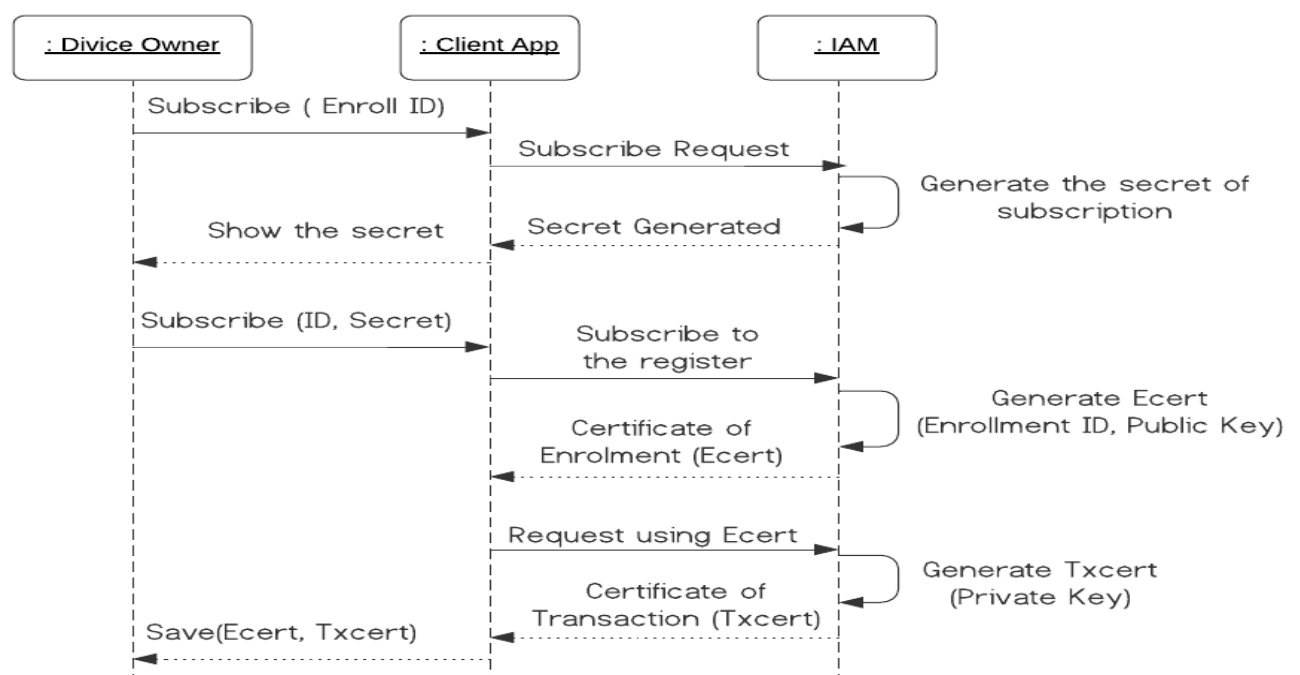


Figure 4.3: Sequence diagram of Identity for blockchain network user

As mentioned above, the third scenario concerns the user who wants to access the blockchain network. Figure 4.4 gives the details.

Owner: this can be a person or an organization that owns the data to be shared between clients (those who have subscribed to the publications) using the MQTT protocol. It can also control access and access to the data by screening applicants.

Worker: This is the passive entity in the system that provides decryption services on behalf of the client. It authenticates new customers with signatures and queries the smart contract for data requested by customers.

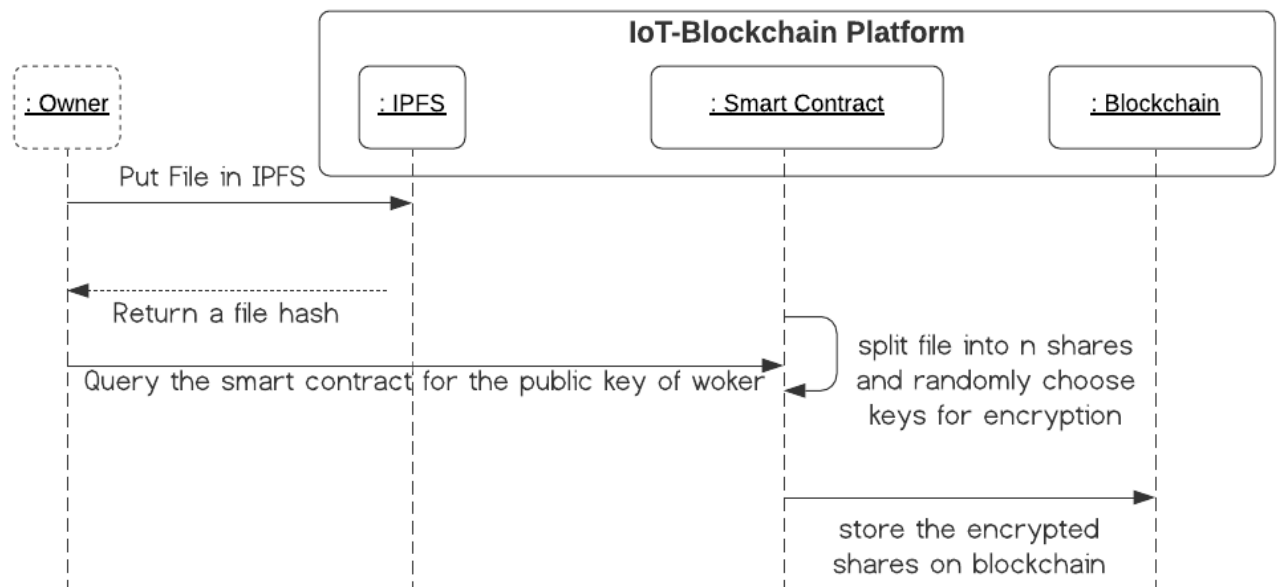


Figure 4.4: Data sharing on IPFS by owner

Figure 4.5 illustrates the different steps to access the data stored in the blockchain through the interplanetary file system (IPFS).

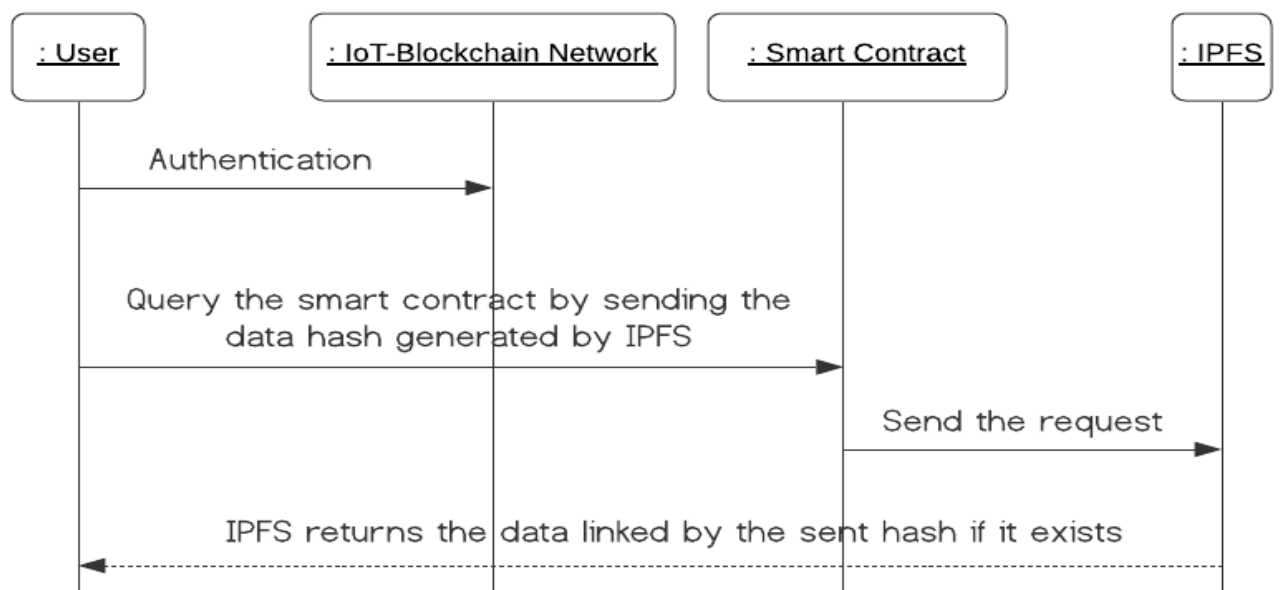


Figure 4.5: Access to the data stored in IPFS

Conclusion

In this part, it was a question of explaining particularly the generalities of the Ethereum blockchain. We gave a brief explanation of how Ethereum works and the differences between it and Bitcoin. We also modeled the different scenarios that result in our architecture, which we proposed in Figure 3.4. Interplanetary File System (IPFS) plays the important role for data storage in the blockchain network. Using the IPFS, large data can be stored and secured in the blockchain without any problems and only their hash will be stored in the distributed ledgers.

5. Implementation of the architecture IoT-Blockchain proposed

In this section, we will present details on the implementation of our architecture that we proposed in Figure 3.4, which is an IoT-Blockchain platform. The proposed system will be based on the private blockchain Ethereum. We know that Ethereum is a distributed platform that is an open source, that efficiently uses solidity, a programmable language that allows to write smart contracts.

The implementation of our system will take two parts. The first one will be based on the different scenarios presented in figure 4.2,4.3 which show how the objects will be able to communicate with each other, send the data in the blockchain and the second one will be based on the storage and downloading of the data in the blockchain network by the IPFS ticket which are moreover presented in figure 4.4,4.5. In both of these two parts, the smart contract will be the key element.

The system consists of a front-end application on the cloud or a local web server, a smart contract deployed on the private blockchain. The private blockchain consists of several P2P nodes connected from different IoT devices such as base stations (BS), the Raspberry Pi which within it contains several sensors and actuators that are connected to it, the laptop or personal computer, etc. The private blockchain can be used to connect to the private blockchain. Each node must install an ethereum client and connect to the same network identifier of the blockchain to participate in the network. When they are part of the same network, they can then execute transactions, make calls, explore and participate in the consensus process.

The smart contract is the business logic of distributed applications (Dapp), and it is not centralized like traditional applications. The contract is deployed on the blockchain by the contract owner using his ethereum account. Once the contract deployment transaction is completed by the miners, it will be available on each node of the network at a specific address. For customers to interact with the blockchain, for example to purchase a ticket, check the validity of the ticket or for the contract owner to change the ticket price, transfer the contract to another account, there is a user-friendly graphical interface. To use the graphical interface, the user must use the Mist browser or the Metamask plugin for chrome and an ethereum account.

Mist or Metamask allow the user to import his ethereum account into the system he is using and will allow the system to connect to the blockchain network by making the user's system a virtual blockchain node. They also inject the web3.js component into the loading HTML pages. Web3.js is a wrapper library that allows data transformation such as encoding and decoding and allows many other features to interact with the RPC port provided by blockchain. If a user uses a direct network node of the blockchain and is able to handle the command line, he does not need Mist or Metamask. The front-end application can be stored on the cloud or on the local web server or the personal computer's web server.

For the first part, we're going to do a little simulation with python. We will create an IoT function that will start to generate the data that will be submitted as a block in the blockchain network. The miners will take care of the validation of the block to be added in the blockchain. The second part is dedicated to the storage and upload of data in the blockchain network through

the IPFS, as explained in the introduction above.

5.1 Blockchain-IoT Simulation

We present a scenario in which a function named IoT generates a string of characters that is considered as a block and is transmitted to the blockchain network for mining. We have also created a function for mining which will have the role of validating the block sent by the IoT function. The consensus applied in our mining function is that of proof of work.

The algorithms we will present below are implemented in python3. We just simulate some system scenarios.

Algorithm 1: IoT function generating the string of characters

```
def IoT(length):
    str = string.ascii_lowercase
    return ''.join(random.choice(str) for i in range(length))
```

The following algorithm is used to mine the strings generated by the IoT function

Algorithm 2: Mining algorithm

```
def mine(self,block):
    for n in range(self.maxNonce) do
        if int(block.hash(),16) <= self.target then
            self.add(block) print(block)
            break
        else
            block.nonce +=1
        end
    end
end
```

We had implemented two classes, one which is in charge of gathering the blocks and the other one is in charge of validating the block to add it in the blockchain once it is validated by the minors following the consensus of the proof of work as shown in figure [5.1](#).

```

k=10
class Block:
    blockNo = 0
    data = random.randint(1,25)
    next = None
    hash = None
    nonce = 0
    previous_hash = 0x0
    timestamp = datetime.datetime.now()
    def __init__(self,data):
        self.data = data

    def hash(self):
        h = hashlib.sha256()
        h.update(
            str(self.nonce).encode('utf-8') +
            str(self.data).encode('utf-8') +
            str(previous_hash).encode('utf-8') +
            str(timestamp).encode('utf-8') +
            str(blockNo).encode('utf-8'))
        return h.hexdigest()
    def __str__(self):
        return "BlockNo: " + str(self.blockNo) + "\ntimestamp: "
            +str(self.timestamp) + "\nBlock Hash: " + str(self.hash()) +"\nNonce: "
            +str(self.nonce)+ "\nData : {}".format(IoT(50)) + "\nDiff: 20"
            +"\nPrevious-hash:" + str(self.previous_hash)

class blockchain:
    diff = 20
    maxNonce = 2**32
    target = 2**(256-diff)
    block = Block1("Genesis")
    dummy = head = block

    def add(self,block):
        block.previous_hash = self.block.hash()
        block.blockNo = self.block.blockNo + 1

        self.block.next = block
        self.block = self.block.next
blockchain = blockchain()
for n in range(k):
    blockchain.mine(Block("Block " + str(n+1)))
    print("-----")
while blockchain.head != None :
    print(blockchain.head)
    blockchain.head = blockchain.head.next

```

Figure 5.1: Python code for simulation Blockchain-IoT

As explained in the third chapter, a blockchain is managed by a network of computers or servers that constantly communicate and exchange information. The computers or servers are called

nodes and form the infrastructure of the blockchain. A full node is a node that maintains a copy of the blockchain. A lightweight node does not maintain a copy of the blockchain and must first connect to a full node before it can interact with the blockchain. As the connected devices encounter many constraints, they will be considered as lightweight nodes. The devices are connected to a full node before it participates in the blockchain network. Here our simulation goes in the same direction, the IoT function, which is considered as a lightweight node, sends the data to the miner function (which is a full node of the network) for validation and saving in the blockchain. Figure 5.2 shows the result of codes shown in Figure 5.1.

Note that all the blocks are linked. Each block contains the hash of the previous block except the first block where the previous hash was generated.

```

BlockNo: 1
timestamp: 2020-06-22 15:32:22.504634
Block Hash: 00000c972f46c20aaa1bb001c25ed8abdaac663361b819217c5e8942cdd3a72f
Nonce: 127890
Data : gymbywptzmgvyvbxarzxogcrxikavloquetuqastzfnmfgozpi
Diff: 20
Previous-hash:3d341db6a7e498d9f7e8c08967f649ae5e29ecca5d7f90c32174f7f9bf9c17d9
-----
BlockNo: 2
timestamp: 2020-06-22 15:32:22.504634
Block Hash: 000007e993c027a3ba052c5537982e39ba6859e6ac9ff408e8fc4d23239df293
Nonce: 2051573
Data : fyoopdcumhpdjvnhvfaeqitauaqhmvdcwfkdxjexyobvgpgttv
Diff: 20
Previous-hash:00000c972f46c20aaa1bb001c25ed8abdaac663361b819217c5e8942cdd3a72f
-----
BlockNo: 3
timestamp: 2020-06-22 15:32:22.504634
Block Hash: 00000ed3d558f27892dae0f6b6a09c518334c93d7fb88d6960c7779cf3500aed
Nonce: 527403
Data : gkbsxrjzzbjgdatbllltudsdccgcqmqicqorkisynxkqlnipgs
Diff: 20
Previous-hash:000007e993c027a3ba052c5537982e39ba6859e6ac9ff408e8fc4d23239df293
-----
BlockNo: 4
timestamp: 2020-06-22 15:32:22.504634
Block Hash: 000007bd11ee51275a5c8b7c40a10aa6590713ffeebf4f8a28c2fc1134851cbb
Nonce: 352315
Data : sbtuulfeuyfjmajhedzslwosyboryonmydnrkgcmjvvypycbsl
Diff: 20
Previous-hash:00000ed3d558f27892dae0f6b6a09c518334c93d7fb88d6960c7779cf3500aed
-----
BlockNo: 5
timestamp: 2020-06-22 15:32:22.504634
Block Hash: 00000e74f183ec2bed6b196d064450cf46b4cd6ffe2e28e1d9b70a4c3d042fe3
Nonce: 1032009
Data : zofojkhhtgretxdxdbksbuwoqfbwxqlhzmgsypulbgkqwdmaxo
Diff: 20
Previous-hash:000007bd11ee51275a5c8b7c40a10aa6590713ffeebf4f8a28c2fc1134851cbb
-----
BlockNo: 6
timestamp: 2020-06-22 15:32:22.504634
Block Hash: 000009da5f3d4457593a3a84e767aa43cc5d48a52f5cd40fe100926b06c73339
Nonce: 1009240
Data : uiwngawrpelozqaughtisrqaccmqbiucgaotdjokcuwwfclflj
Diff: 20
Previous-hash:00000e74f183ec2bed6b196d064450cf46b4cd6ffe2e28e1d9b70a4c3d042fe3
-----

```

Figure 5.2: Blockchain

5.2 Data storage in the IPFS

As we explained in previous chapters, storing data in the blockchain poses serious problems because the size of a block is limited. Hence we added IPFS technology to overcome this problem. IPFS means InterPlanetary File System. It is a peer-to-peer distributed file system that provides distributed storage to network peers (Chen et al., 2017). The fundamental data structure used by IPFS is Merkle DAG. A similar data structure is also used in distributed version control systems such as Git. Peer nodes do not need to trust each other to store and access data on the IPFS network. IPFS works on content-addressed storage, whereby data stored on IPFS will be linked to the cryptographic hash of its content. IPFS uses a blockchain to store IPFS links. The data itself is therefore not stored in the blockchain. IPFS is similar to BitTorrent, Distributed Hash Table, Git, Self-Certified File Systems, because it reuses similar concepts. IPFS provides a decentralized way of storing and referencing files, but gives you more control and refers files by hashing. There are no peer node privileges in IPFS, so there is no concept of a master or leader node. One of the advantages of IPFS over HTTP is that it offers higher throughput when processing large chunks of data. IPFS supports a decentralized architecture, unlike HTTP which supports a client-server architecture. IPFS generally runs on port number 5001.

IPFS and the Blockchain are a perfect match. Large datasets can be processed with IPFS and the immutable and permanent IPFS links can be placed in a blockchain transaction. This timestamps and secures the content, without having to put the data on the chain itself.

Let's try to put a simple distributed application (DApp) to download a document on IPFS, then store the IPFS hash on the Ethereum blockchain. Once the IPFS hash number is sent to the Ethereum blockchain, the user will receive a transaction receipt.

We present the scenario where the user or device owner downloads a file on IPFS by invoking an Ethereum contract. The hash of the file will be stored on Ethereum.

Here is the process we will follow:

- Taking the file as input
- Convert the file to a buffer
- Loading the buffer on IPFS
- Store the hash of the file returned by IPFS
- Get the user's Metamask Ethereum address
- The user confirms the transaction with Ethereum via Metamask
- IPFS hash is written on Ethereum

To manage the Ethereum nodes in their cloud, we are going to use Infura's technology which is access to the API of the Ethereum network and the IPFS.

5.3 Use of Infura and IPFS on the Ethereum blockchain

In some cases, Infura and/or IPFS have come to solve many of the problems encountered by the blockchain such as : It is expensive to store data on the Ethereum blockchain, It is difficult to configure a geth Ethereum client, It is difficult to upgrade the infrastructure.

By using Infura, access to the Ethereum network and IPFS becomes much faster. Synchronizing the geth client no longer takes hours, which uses a huge amount of memory and bandwidth while the entire blockchain is downloaded.

Here are some other advantages of using Infura:

- Huge quantities of data can be stored on IPFS, and only the file hash can be stored on Ethereum;
- Infura provides secure, reliable, scalable and easy-to-use APIs for accessing the Ethereum network and IPFS;
- Infura provides TLS-compatible public endpoints;
- The code is portable to the Ethereum interface using JSON RPC, Web3, etc..

5.3.1 Technology to be used.

- React: Front Library;
- Solidity: The language used to build smart contracts that works on Ethereum;
- IPFS: Decentralized storage;
- Infura: API access to the Ethereum network and IPFS.

5.3.2 Local machine.

The test machine validating the mining operations is a laptop computer with a Core i5-8G Intel®processor Core™ i5-8250U CPU @1.60GHz8 and 8 GB RAM, Intel®UHD Graphics 620 (KabyLake GT2), GNOM 3.28.2, Ubuntu 18.04 64-bit operating system type.

Before being able to start the server that will be the local machine in our case, we will first have to install some dependencies to better prepare the environment of the Ethereum blockchain and IPFS :

```
npm i -g create-react-app, npm install react-bootstrap, npm install fs-extra, npm install ipfs-api, npm install web3
```

Due to testing, we had deployed our smart contract at remix.ethereum which is a platform for the development and deployment of smart contract of ethereum. In order to deploy the smart

contract we need ether, hence the use of the Ropsten Testnet network appears useful to us because it offers us ether for testing.

Algorithm 3: Pseudo code for the Smart contract

```
Input : File;
ipfsHash = File;
while ipfsHash = True do
    sendFile = ipfsHash;
    if sendFile = True then
        | Return Hash(sendFile);
    else
        | Go to input File;
    end
end
```

To develop a web page that will be able to interact on the Ethereum blockchain, we need to use a javascript library called web3.js. This library allows to enter the address of the smart contract and to call the functions it contains, by passing possibly the necessary parameters.

Algorithm 4: Pseudo code for storage of the file hashed returned by IPFS

```
Input : ipfsFileHash;
AddrUser = GetMetamaskEthereumAddress;
while AddrUser = True do
    if ipfsFileHash = True then
        | UserconfirmTransction;
        | StoreTransOnEthereum;
    else
        | Reinsert MetamaskEthereumAddress;
        | Repeat the second step ;
    end
end
```

A smart contract is a self-executing script that can be used to interface with the blockchain. In our example, this is our way of adding and receiving IPFS addresses on the Ethereum blockchain. Figure 5.3 shows the contract in our example written in solidity and deployed using Remix on the Rinkeby testnet.



Figure 5.3: Smart contract written in solidity

Figure 5.4 shows the implementation of the storehash class that contains the contract address and the contract application binary interface (ABI) that is the standard way to interact with contracts in the Ethereum ecosystem, both from outside the blockchain and for contract-to-contract interaction.

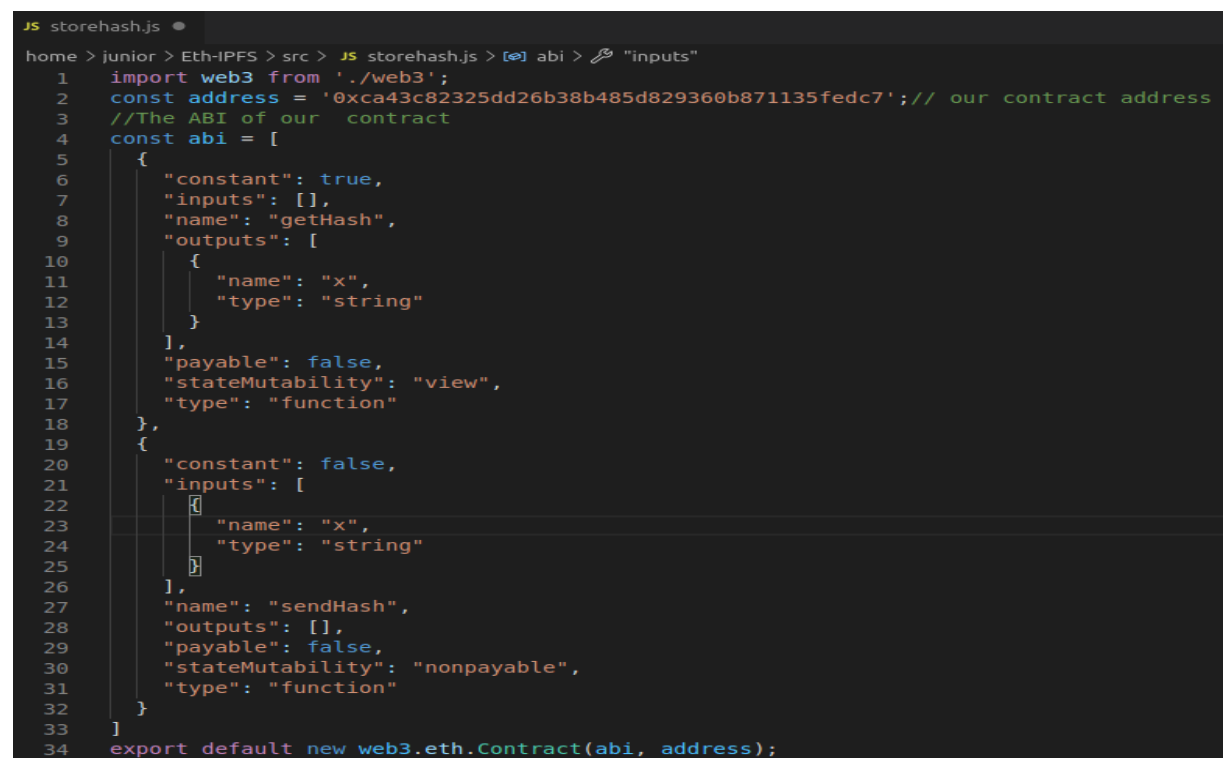
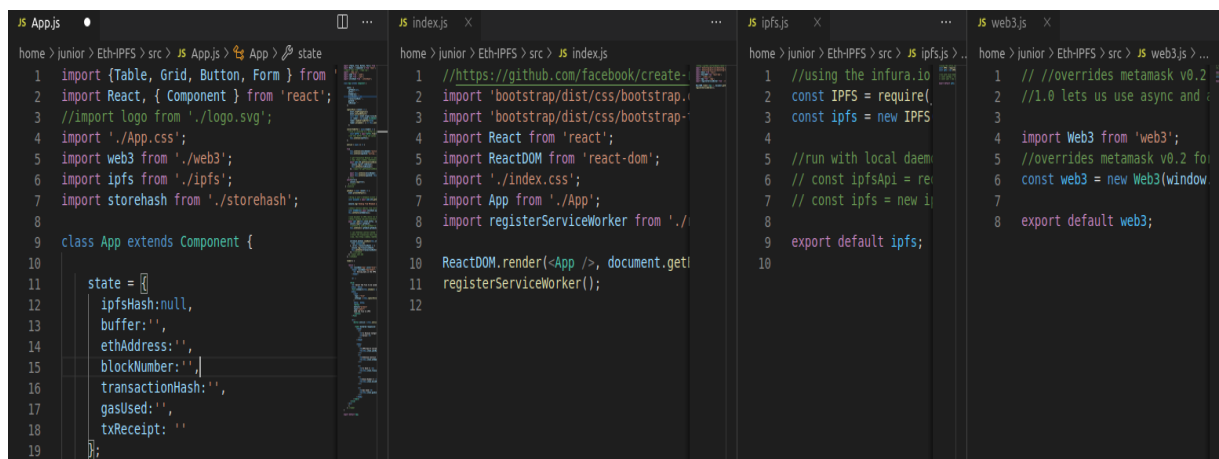


Figure 5.4: ABI and Address of the contract

Figure 5.5 shows the different layers that contributed to the realization of our program that allows a user to download a document on IPFS and then store the IPFS hash on the Ethereum blockchain. Once the IPFS hash number has been sent to the Ethereum blockchain, the user will receive a transaction receipt.



```
JS App.js
home > junior > Eth-IPFS > src > JS App.js > App > state
1 import {Table, Grid, Button, Form} from 'react-bootstrap';
2 import React, { Component } from 'react';
3 //import logo from './logo.svg';
4 import './App.css';
5 import web3 from './web3';
6 import ipfs from './ipfs';
7 import storehash from './storehash';
8
9 class App extends Component {
10
11   state = {
12     ipfsHash:null,
13     buffer:'',
14     ethAddress:'',
15     blockNumber:''
16   };
17   transactionHash:'',
18   gasUsed:'',
19   txReceipt:''
20 };
21
22 JS index.js
home > junior > Eth-IPFS > src > JS index.js
1 //https://github.com/facebook/create-react-app
2 import 'bootstrap/dist/css/bootstrap.min.css';
3 import 'bootstrap/dist/css/bootstrap.min.css';
4 import React from 'react';
5 import ReactDOM from 'react-dom';
6 import './index.css';
7 import App from './App';
8 import registerServiceWorker from './registerServiceWorker';
9
10 ReactDOM.render(<App />, document.getElementById('root'));
11 registerServiceWorker();
12
13 JS ipfs.js
home > junior > Eth-IPFS > src > JS ipfs.js
1 //using the infura.io
2 const IPFS = require('ipfs-http-client');
3 const ipfs = new IPFS({host:'infura.io',port:5001});
4
5 //run with local daemon
6 // const ipfsApi = 'http://127.0.0.1:5001';
7 // const ipfs = new IPFS({host:ipfsApi,port:5001});
8
9 export default ipfs;
10
11 JS web3.js
home > junior > Eth-IPFS > src > JS web3.js
1 //overrides metamask v0.2
2 //1.0 lets us use async and await
3
4 import Web3 from 'web3';
5 //overrides metamask v0.2 for web3
6 const web3 = new Web3(window.web3.currentProvider);
7
8 export default web3;
```

Figure 5.5: Different layers of the application

Figure 5.6 shows the execution of the program on a browser with which Metamask is installed. The user will download a file that will be stored on IPFS.

localhost:3000

Storing data in the IPFS using the Ethereum blockchain

Select the file to be saved to the IPFS :

Choisir un fichier | Aucun fichier choisi

Send the file to IPFS

Receive a Transaction record

Tx Receipt Category	Values
IPFS Hash # stored on Eth Contract	
Ethereum Contract Address	
Tx Hash #	

Storing data in the IPFS using the Ethereum blockchain

Select the file to be saved to the IPFS :

Choisir un fichier | Pass_JUNIOR_c.pdf

Send the file to IPFS

Receive a Transaction record

Tx Receipt Category	Values
IPFS Hash # stored on Eth Contract	Qmd3dgnhNfsVTsLcj28YEhqz12otsTrHFZGRyYScw22qhw
Ethereum Contract Address	0x61E991F0b59248570187aE1a24B26184A5d6c18B
Tx Hash #	

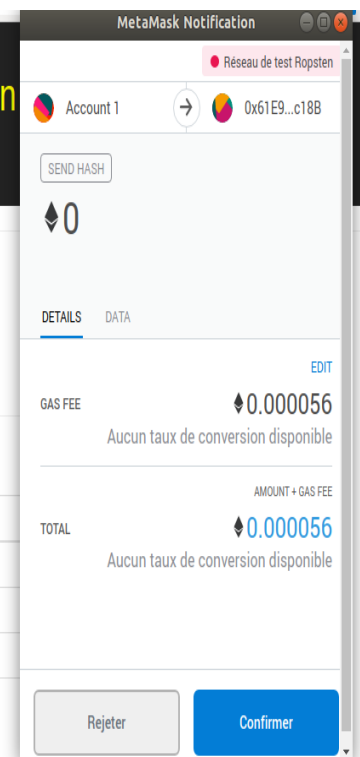


Figure 5.6: User's Interface

After confirming the transaction via Metamask which is a crypto wallet that can be used on Chrome, Firefox and Brave browsers. It allows to make the link between a classic browser and the Ethereum blockchain. It is also a browser extension that allows to browse the Ethereum blockchain. The hash that has been generated by IPFS will be automatically saved in the Ethereum blockchain. The IPFS hash starts with the prefix Qm... and it is the last one that will be stored in the blockchain.

Figure 5.7 shows the hash generated by IPFS, the address of our smart contract and the hash of the transaction. This is the receipt sent by IPFS and Ethereum.

Select the file to be saved to the IPFS :

Pass_JUNIOR_c.pdf

Tx Receipt Category	Values
IPFS Hash # stored on Eth Contract	Qmd3dgnhNfsVTsLcj28YEhqz12otsTrHFZGRyYScw22qhw
Ethereum Contract Address	0x61E991F0b59248570187aE1a24B26184A5d6c18B
Tx Hash #	0xaa2023f5dac42c8f5ccd69e6cb38701a2513374bb25214a2457085de98c0e902

Figure 5.7: IPFS's Transaction on Ethereum blockchain

Since we already have our receipt and we are connected to an IPFS node via localhost: 3000, we can now see our file that we stored in the IPFS on this IPFS gateway:

<https://gateway.ipfs.io/ipfs/> + the IPFS hash#

So, let's see the file we registered:

<https://gateway.ipfs.io/ipfs/Qmd3dgnhNfsVTsLcj28YEhqz12otsTrHFZGRyYScw22qhw>
which is shown in figure 5.8



In this chapter, we presented the simulation of a scenario of a connected object that sends data to be stored in the blockchain. For this, we had created an IoT function in python that simulates the role of a connected object and a function for mining. The mining function plays the role of the minor nodes that participate in the validation of a block and to add it in the blockchain. In the second part, we showed how to save the data in the IPFS which is the decentralized storage system. It stores the data and generates the hash that will be recorded in the Ethereum blockchain.

Conclusion and Perspective

Here we are at the end of our research which was based on the implementation of intelligent and light cryptographic systems to reinforce trust and security in the Internet of Things. The IoT revolution is based on connected objects as a source of data throughout their life cycle, focusing on permanent connectivity and the systematic exploitation of this data through Bigdata. The Internet of Things represents a global ideology that removes the boundaries between the physical and virtual worlds, and covers most computer systems and information technologies. Indeed, the IoT encompasses several domains such as wireless sensor networks, smart cities, connected vehicles, control systems, connected industry and so on. Therefore, an IoT system is a heterogeneous system using different technologies, deployed on different architectures and platforms and implemented on a wide variety of computing materials. Usually, it uses wireless communication technologies to connect intelligent and autonomous objects. These objects have the ability to gather, analyze, process, generate and exchange information in order to provide advanced services. However, computer security issues significantly slow down the rapid evolution and deployment of this advanced technology.

In this thesis, we proposed the use of the elliptic curve cryptographic protocol to guarantee data confidentiality during exchanges between objects within the network. The choice on this protocol is related to the fact that the connected objects meet several constraints such as computing power, energy and storage. A light cryptographic protocol that can be adapted to these objects should be defined to overcome these various constraints. We have also proposed an architecture combining the Internet of Things and blockchain technology. The main advantage of blockchain technology is that it allows objects to trust each other without the intervention of a “trusted third party” for their communication. This technology guarantees security and privacy because anonymity is one of the basic principles of the blockchain. The success of the IoT indeed passes through the blockchain and its infrastructure of distributed algorithmic trust. With blockchain, the fact that a transaction is accepted or rejected is the result of a distributed consensus and not of a centralized institution. Consensus is at the heart of the blockchain business model, through the use of smart contracts, which are irrevocable computer programs, most often deployed on a blockchain, that execute a set of pre-defined instructions. We have opted to use the Ethereum blockchain because of its adaptability to the IoT environment. Knowing that the blockchain does not store large amounts of data, we combined our architecture with the Interplanetary File System (IPFS) for data storage to overcome this problem. IPFS returns the hash of the data to be stored and this is the hash that will be stored in the blockchain.

Perceptually, we will improve the architecture of our system by integrating new features for optimizing communication and data transmission between objects. The connected objects form a complex network, from which we will have to define some parameters to detect the objects belonging to the same community and this will help us to know for example the distribution degree, the eccentricity, etc... of each node participating in the network.

References

- Chafiq Titouna Ousmane Thiare Alidou Mohamadou Abdelhak Mourad Gueroui et al. Ado Adamou Abba Ari, Olga Kengni Ngangmo. Enabling privacy and security in cloud of things: architecture, applications, security & privacy challenges, applied computing and informatics. 2019.
- Tahir Ahmad and Silvio Ranise. Validating requirements of access control for cloud-edge iot solutions (short paper). In *International Symposium on Foundations and Practice of Security*, pages 131–139. Springer, 2018.
- Marah Nana Awa Chafiq Titouna Nabila Labraoui Joseph Yves Effa Wahabou Abdou Arouna Ndam Njoya, Ado Adamou Abba Ari and Abdelhak Gueroui. Hybrid wireless sensors deployment scheme with connectivity and coverage maintaining in wireless sensor networks. *Wireless Personal Communications*, 112(80):1–25, 2020.
- Elaine Barker, William Barker, William Burr, William Polk, Miles Smid, et al. *Recommendation for key management: Part 1: General*. National Institute of Standards and Technology, Technology Administration, 2006.
- Djehina Boukara. Utilisation des ontologies pour l'intégration d'internet des objets dans la gestion des processus métier. 2019.
- CByothe.fr. Internet des objets, 2020. URL www.byothe.fr/2018/10/histoire-internet-des-objets-iot/.
- Shi-Cho Cha, Jyun-Fu Chen, Chunhua Su, and Kuo-Hui Yeh. A blockchain connected gateway for ble-based devices in the internet of things. *IEEE Access*, 6:24639–24649, 2018.
- Yongle Chen, Hui Li, Kejiao Li, and Jiyang Zhang. An improved p2p file system scheme based on ipfs and blockchain. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2652–2657. IEEE, 2017.
- Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. CRC press, 2005.
- Connectwave. Les fonctions de l'objet connecté, 2020. URL www.connectwave.fr/techno-appli-iot/iot/reseaux-et-infrastructures-iot/.
- Renaud Dumont. Cryptographie et sécurité informatique. *Eyrolles*, 2010, 2009.
- H. Khemissa et D. Tandjaouig. A lightweight authentication scheme for e-health applications in the context of internet of things. In *In 2015 9th International Conference on Next Generation Mobile Applications*, page 90–95. Services and Technologies, 2015.
- ethdocs. Connecting to the network, 2020. URL <https://ethdocs.org/en/latest/network/connecting-to-the-network.html>.

- Youssou Faye. *Algorithmes d'authentification et de cryptographie efficaces pour les réseaux de capteurs sans fil*. PhD thesis, 2014.
- framboiseaupotager.blogspot. Raspberry pi episode 3, 2020. URL <http://framboiseaupotager.blogspot.com/2018/09/raspberry-pi-episode-3-booter-sur-un.html>.
- Tianhe Gong, Haiping Huang, Pengfei Li, Kai Zhang, and Hao Jiang. A medical healthcare system for privacy protection based on iot. In *2015 Seventh International Symposium on Parallel Architectures, Algorithms and Programming (PAAP)*, pages 217–222. IEEE, 2015.
- Ranida Hamidouche, Zibouda Aliouat, Abdelhak Mourad Gueroui, Ado Adamou Abba Ari, and Lemia Louail. Classical and bio-inspired mobility in sensor networks for iot applications. *Journal of Network and Computer Applications*, 121:70–88, 2018.
- Badis Hammi, Achraf Fayad, Rida Khatoun, Sherali Zeadally, and Youcef Begriche. A lightweight ecc-based authentication scheme for internet of things (iot). *IEEE Systems Journal*, 2020.
- Lei Hang and Do-Hyeun Kim. Design and implementation of an integrated iot blockchain platform for sensing data integrity. *Sensors*, 19(10):2228, 2019.
- Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.
- Tobias Heer, Oscar Garcia-Morchon, René Hummen, Sye Loong Keoh, Sandeep S Kumar, and Klaus Wehrle. Security challenges in the ip-based internet of things. *Wireless Personal Communications*, 61(3):527–542, 2011.
- D.Tandjaouig H.Khemissa. A lightweight authentication scheme for e-health applications in the context of internet of things. *9th International Conference on Next Generation Mobile Applications*, 121:90–95, 2015.
- Farah Khedim, Nabila Labraoui, and Ado Adamou Abba Ari. A cognitive chronometry strategy associated with a revised cloud model to deal with the dishonest recommendations attacks in wireless sensor networks. *Journal of Network and Computer Applications*, 123:42–56, 2018.
- Frederic Lemoine. *Internet des Objets centré service autocontrôlé*. PhD thesis, 2019.
- LesJeudis. 10 applications ido, 2020. URL <https://blog.lesjeudis.com/10-applications-de-l-internet-des-objets-qui-revolutionnent-la-societe>.
- Ming Li, Shucheng Yu, Kui Ren, and Wenjing Lou. Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings. In *International conference on security and privacy in communication systems*, pages 89–106. Springer, 2010.
- Alfred J Menezes, Jonathan Katz, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

- Muhammad Faheem Mushtaq, Sapiee Jamel, Abdulkadir Hassan Disina, Zahraddeen A Pindar, N Shafinaz Ahmad Shakir, and Mustafa Mat Deris. A survey on the cryptographic encryption algorithms. *International Journal of Advanced Computer Science and Applications*, 8(11): 333–344, 2017.
- Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203, 2007.
- A.O.Mulani P.B.Mane. High speed area efficient fpga implementation of aes algorithm. *AISSMS Inst.of Information Technology*, page 158, 2018.
- Harsha S Gardiyawasam Pussewalage and Vladimir Oleshchuk. A patient-centric attribute based access control scheme for secure sharing of personal health records using cloud computing. In *2016 IEEE 2nd International Conference on Collaboration and Internet Computing (CIC)*, pages 46–53. IEEE, 2016.
- Ammar Rayes and Samer Salam. Internet of things from hype to reality. *The Road to Digitization; River Publisher Series in Communications; Springer: Basel, Switzerland*, 49, 2017.
- Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005.
- Somia SAHRAOUI. *Mécanismes de sécurité pour l'intégration des RCSFs dans l'IoT (Internet of Things)*. PhD thesis, Université Mustapha Ben Boulaid Batna 2, Département de l'informatique, 2017.
- William Stallings. Intruders. *Cryptography and Network Security, Fourth Edition, Prentice Hall*, pages 565–594, 2005.
- tutorialspoint. Cryptography digital signatures, 2020. URL https://www.tutorialspoint.com/cryptography/cryptography_digital_signatures.htm.
- Binod Vaidya, Min Chen, and Joel JPC Rodrigues. Improved robust user authentication scheme for wireless sensor networks. In *2009 Fifth International Conference on Wireless Communication and Sensor Networks (WCSN)*, pages 1–6. IEEE, 2009.
- Ludovica Visciola. Digitalization in medicine: the internet of medical things (iomt).
- wikipedia. Chiffrement par décalage, 2020. URL https://fr.wikipedia.org/wiki/Chiffrement_par_d%C3%A9calage.
- UIT-T X.1361. Cadre de sécurité applicable à l'internet des objets fondé sur le modèle passerelle, 09/2018.
- Youssef Ould Yahia. *Proposition d'un modèle de sécurité pour la protection de données personnelles dans les systèmes basés sur l'internet des objets*. PhD thesis, 2019.