République Algérienne Démocratique et Populaire
Université Abou Bekr Belkaid -Tlemcen

Faculté des Sciences

Département d'Informatique

Mémoire de Fin d'Études

Pour l'obtention du diplôme de Master en Informatique
Option : **Réseaux et Systèmes Distribués (R.S.D)**

Thème

Implementation of an access control protocol in an IoT environment

---

Réalisé par :

LOURIACHI Mohammed Abderrahmane

Soutenu le **02 Juillet 2025** devant le jury composé de :

Mr SAIDI Abdessamad	Président
Mr LEHSAINI Mohamed	Encadrant
Mr DEGDEG Hicham	Co-encadrant
Mr SEBBAH Abderrazzak	Examinateur
Mr ETCHIALI Abdelhak	Expert I2E

Année universitaire : 2024/2025

بسم الله الرحمان الرحيم

الحمد لله الذي بِنِعْمَتِهِ تتم الصالحات وبشكره تدوم النعم

Acknowledgement

First of all, I would like to thank Almighty God who gave me patience and strength to accomplish this work.

I am deeply grateful to my family, especially my beloved mother.May Allah protect her, bless her, and grant her lasting health and happiness.

I would like to sincerely thank Mr. **LEHSAINI Mohamed** and Mr. **DEGDEG Hichem**, who supervised this project. I am very grateful for their help, advice, and insightful comments.

I would like to thank all those who kept us in their prayers. I ask Allah to fill their homes with happiness and blessing.

Finally, our heartfelt thanks go to the members of the jury for their interest in this project, for agreeing to examine this work and enriching it with their experiences

Dedication

I dedicate this work to my beloved mother, whose sacrifices and love have been the foundation of my happiness and success.

To my father, one of the greatest men I have ever known. To my brothers—may Allah grant them success and guide them toward the work they seek.

To my friends in the neighborhood: Akram, Hamza, Mounim, Alaa, and Saddik.

And to my friends at the university: Majdoub, Bouchour, Krifif, and Rahoui.

Abstract

This final-year project explores the implementation of access control in industrial environments where IoT objects are used to operate and monitor machines. The approach adopted for this system is based on decentralised access control. This is guaranteed by the use of an attribute-based cryptographic system, represented by the MA-ABE approach (Multi-Authority Attribute Based Encryption). The result is a platform that enables an organisation's authorities to control user access to IoT objects. This demonstrates the viability of cryptography as a means of stateless access control.

Keywords: IoT , ABE , MA-ABE , Access control , Decentralised access control

Résumé

Ce projet de fin d'études explore la mise en œuvre du contrôle d'accès dans les environnements industriels où les objets IoT sont utilisés pour faire fonctionner et surveiller les machines. L'approche adoptée pour ce système est basée sur un contrôle d'accès décentralisé. Celui-ci est garanti par l'utilisation d'un système cryptographique basé sur les attributs, représenté par l'approche MA-ABE (Chiffrement multi-autorités basé sur les attributs). Le résultat est une plateforme qui permet aux autorités d'une organisation de contrôler l'accès des utilisateurs aux objets IoT. Cela démontre la viabilité de la cryptographie comme moyen de contrôle d'accès sans état.

Mots clés: IoT, ABE, MA-ABE, Contrôle d'accès, Contrôle d'accès décentralisé

ملخص

يستكشف مشروع التخرج هذا تنفيذ التحكم في الوصول في البيئات الصناعية حيث يتم استخدام أجهزة إنترنت الأشياء لتشغيل الآلات ومراقبتها. يعتمد النهج المتبع في هذا النظام على التحكم في الوصول اللامركزي. ويتم ضمان ذلك من خلال استخدام نظام تشفير قائم على السمات، والذي يمثله نهج MA-ABE. وتكون النتيجة عبارة عن منصة تسمح لسلطات المنظمة بالتحكم في وصول المستخدم إلى أجهزة إنترنت الأشياء. يوضح هذا مدى جدوى التشفير كوسيلة للتحكم في الوصول بدون حالة.

الكلمات المفتاحية:
إنترنت الأشياء، ABE، MA-ABE، التحكم في الوصول، التحكم في الوصول اللامركزي

CONTENTS

General Introduction **1**

1 Concepts of IoT security **3**

1.1 Introduction . . . 3
1.2 IoT security . . . 4
1.3 Fundamentals of IoT security . . . 4
1.4 Access control mechanisms . . . 5
1.4.1 Discretionary access control (DAC) . . . 5
1.4.2 Mandatory access control (MAC) . . . 5
1.4.3 Role-based access control (RBAC) . . . 6
1.4.4 Attribute-based access control (ABAC) . . . 6
1.4.5 Multi-authority attribute-based access control (MA-ABE) . . . 7
1.5 Resource-limited devices . . . 7
1.6 Attack vectors on IoT devices . . . 8
1.7 Computing cost of encryption . . . 9
1.8 IoT communication protocols . . . 10
1.8.1 IoT communication over http . . . 10
1.8.2 MQTT protocol . . . 10
1.8.3 CoAP protocol . . . 10
1.9 Conclusion . . . 12

2 Architecture of the proposed platform **13**

2.1 Introduction . . . 13
2.2 Objectives of the platform . . . 13
2.3 MA-ABE and access control . . . 14
2.3.1 Identity-based encryption (IBE) . . . 14
2.3.2 Attribute-based encryption (ABE) . . . 14
2.3.3 Multi Authority Attribute Based Encryption (MA-ABE) . . . 15
2.4 Integration of fog computing into the platform . . . 18

2.5 Specification of system requirements . . . 18
2.5.1 Identification of actors . . . 18
2.5.2 Functional requirements . . . 19
2.5.3 Non-functional requirements . . . 19
2.6 The design of the system . . . 20
2.6.1 User case diagram . . . 20
2.6.2 Sequence diagram . . . 21
2.6.2.1 Various authority scenarios . . . 21
2.6.2.2 Admin scenarios . . . 24
2.6.2.3 Fog node scenarios . . . 26
2.6.2.4 IoT device scenarios . . . 27
2.6.2.5 Various user scenarios . . . 28
2.6.3 Class diagram . . . 29
2.7 Architecture of the proposed system . . . 31
2.8 Conclusion . . . 33

3 Development and Implementation of an IoT Access Control System 34
3.1 Introduction . . . 34
3.2 Conception Tools . . . 34
3.2.1 Draw.io . . . 34
3.2.2 Figma . . . 35
3.3 Hardware tools . . . 35
3.4 Software tools . . . 36
3.4.1 Arduino IDE . . . 36
3.4.2 Android Studio . . . 36
3.4.3 Node.js . . . 37
3.4.4 GoLang . . . 37
3.4.5 Database . . . 37
3.4.6 Used libraries . . . 37
3.4.6.1 MA-ABE library . . . 37
3.4.6.2 AIOCoAP protocol . . . 38
3.4.6.3 CoAP-simple-library . . . 38
3.4.6.4 Californium . . . 39
3.5 Software use in the system . . . 39
3.6 Presentation of the application . . . 39
3.6.1 Admin interface . . . 40
3.6.2 Authority interface . . . 42
3.6.3 User's application . . . 45
3.7 Scenario-based testing . . . 47
3.7.1 Initializing the fog node . . . 47
3.7.2 Adding IoT objects to the Fog node . . . 49

3.7.3 Connecting a user to IoT objects 50
3.8 Conclusion 50

General Conclusion **51**

Bibliography **52**

Business Model Canvas (BMC) **56**

LIST OF FIGURES

1.1 Access control model (DAC) [5] 5
1.2 Mandatory access control (MAC) [6] 6
1.3 Access control model (RBAC) [7] 6
1.4 Access control model (RBAC) [8] 7
1.5 Attack vectors on IoT devices 9

2.1 Illustration of the ABE approach 14
2.2 Difference between CP-ABE and KP-ABE 15
2.3 Example of Multi-Authority ABE 16
2.4 Encryption and decryption in MA-ABE 17
2.5 Incorporation fog computing in the platform 18
2.6 User case diagram of the platform proposed 20
2.7 Sequence diagram for Establishment of a new authority 21
2.8 Sequence diagram for Adding new attribute 22
2.9 Sequence diagram for Renewing attribute keys 22
2.10 Sequence diagram for Adding a new user 23
2.11 Sequence diagram for Adding IoT devices 25
2.12 Sequence diagram for Fog node scenarios 26
2.13 Sequence diagram for IoT device registration 27
2.14 Sequence diagram for IoT object receiving a request from a user 28
2.15 Sequence diagram for User requesting access to an IoT device 29
2.16 Class diagram of the proposed security system 30
2.17 Architecture of the proposed system 31
2.18 The two phases of the system: set-up and operational 32
2.19 The use of the Key Derivation Function (KDF) 32

3.1 Raspberry PI 4 model B 36
3.2 NodeMCU 36
3.3 Process of manipulating the authorities 38

3.4 Creation of JSON utility functions 38
3.5 Interaction between the admin and the authorities 39
3.6 Relationship between the admin, the user application and the IoT objects 39
3.7 Home page of the admin 40
3.8 List of fog nodes page 40
3.9 Fog node page 41
3.10 Authorities page 42
3.11 Users page 42
3.12 Authority interface 43
3.13 Attribute handling page 43
3.14 Import of public parameters 44
3.15 Creation of new authority page 44
3.16 Users handling page 45
3.17 Login page for the user's application 45
3.18 Home page 46
3.19 Authorities page 46
3.20 IoT object list page 47
3.21 Fog node initialization via *"./install.sh"* 48
3.22 Starting the Fog node using python3 *"server.py"* 48
3.23 Page for adding an IoT object 49
3.24 Token request by IoT object at Fog node 49
3.25 Generation of a token by the Fog node 49
3.26 View the object via the application 50
3.27 Access to the IoT object 50

LIST OF TABLES

1.1 Specifications for two widely used IoT microcontrollers and the Raspberry Pi-4 [11, 12, 13] 8
1.2 Comparison between CoAP and HTTP 11
1.3 Comparison between OSCoRE and DTLS 12

ACRONYMS

ABAC Attribute-Based Access Control. 6

ABE Attribute-based encryption. 7, 14

AEAD Authenticated Encryption with Associated Data. 11

AIOCoAP Asynchronous IO Concurrent Application Protocol. 38

Arduino IDE Arduino Integrated Development Environment. 36

CoAP Constrained Application Protocol. 10

CP-ABE Cyphertext policy attribute based encryption. 15

DAC Discretionary access control. 5

DDoS distributed denial of service. 4

DoS denial of service. 4

DTLS Datagram Transport Layer Security. 10, 11

E2E End-to-End Encryption. 11

EDHOC Ephemeral Diffie-Hellman Over COSE. 11

GoFE GO Functional Encryption. 37

HKDF HMAC-based Key Derivation Function. 11

HTTP Hypertext Transfer Protocol. 8, 10

IBE Identity-based encryption. 14

IIoT Industrial Internet of Things. 5

IoT Internet of Things. 1, 3

KP-ABE Key Policy Attribute Based Encryption. 15

MA-ABE Multi-Authority Attribute Based Encryption. 1, 7, 13, 15

MAC Mandatory access control. 5

MiTM man-in-the-middle. 4, 8, 14

MQTT Message Queuing Telemetry Transport. 10

MySQL (My Structured Query Language. 37

OSCoRE Object Security for Restricted RESTful Environments. 11

PKI public key infrastructure. 14

PSK Pre-Shared Key. 11

RBAC Role-based access control. 6

TCP Transmission Control Protocol. 10

TLS Transport Layer Security. 10

TLS/SSL Transport Layer Security and Secure Sockets Layer. 8

UDP User Datagram Protocol. 10

UML Unified Modeling Language. 2, 34

GENERAL INTRODUCTION

General Introduction

Internet of Things (IoT) has become an essential foundation of modern industry, playing a major role in production in smart factories and automated systems. By connecting physical devices such as sensors, machines and controllers to the Internet, IoT enables industries to monitor, collect and analyse data in real time. This enables better decisions to be made, efficiency to be improved and machines to be better controlled. Although an IoT novice might think of IoT devices as ordinary computers, they range from small microcontrollers to full-sized motherboards. These devices operate autonomously and communicate over open networks, making them particularly vulnerable to a wide range of security breaches and cyber threats.

In the context of industrial environments such as factories, warehouses and businesses, the need for robust access control and security mechanisms becomes even more important. These environments typically involve a high density of sensors, control systems and monitoring units, which need to interact securely and reliably. Unauthorised access to these systems will lead to disruptions, security risks and data breaches.

The aim of this project is to meet these challenges by proposing the design and implementation of a platform called *"SmarDustry"*. This platform guarantees security and access control in industrial sectors. It takes into account the structure and organisation of factories and companies, enabling hierarchical access management with authorities within the organization able to control access independently of each other, thus reflecting the organizational structure of industrial companies. To achieve this, we have implemented the Multi-Authority Attribute Based Encryption (MA-ABE) security mechanism to guarantee enhanced system security.

The document is divided into four chapters.

- Chapter 1 introduces the Internet of Things and analyzes the security of IoT devices, focusing on the challenges they face, the attack vectors and the protocols used in IoT security and communications.
- Chapter 2 discusses the design of the platform enabling security and access control in an industrial sector, presents the main objectives of the platform, explain the technologies

used to achieve them and details the design of the platform using the Unified Modeling Language (UML) to improve understanding of the system.

- Chapter 3 presents the technologies, software and materials used.
- Chapter 4 demonstrates the complexity of the platform by running scenarios.

Finally, we conclude with a summary of the project and a discussion of potential future work and improvements that could enhance the platform's functionality.

CHAPTER 1

CONCEPTS OF IOT SECURITY

1.1 Introduction

IoT includes a wide range of devices such as sensors, appliances and machines that can interact remotely over the network. They vary in size, cost, sensors used, computing power and application. Communication to and from IoT devices is divided into two categories.

- Telemetry: this is the aggregation of data generated by sensors and sending it to a server.
- Remote control: this is the control of IoT devices by sending them commands.

IoT is divided into four layers [1]:

- **Perception layer** : This layer interacts with the physical environment to collect raw data. Devices connected to the IoT such as sensors and cameras passively collect information and images to be communicated via the network layer, while actuators instruct devices to perform tasks based on sensor data or additional commands in IoT systems. Actuators are hardware devices that convert energy into motion and operate on the environment.
- **Network layer**: This layer is responsible for the flow and transfer of data between the sensors in the perception layer and the processing layer through various networks, for example the transfer of data between IoT devices and back-end systems using WiFi, Bluetooth, etc.
- **Data Processing layer**: The data processing layer, sometimes called the middleware layer, stores, analyzes and pre-processes data from the network layer. This includes activities such as data aggregation, protocol translation and the application of security measures to data ready for the application layer. In addition, message brokers, IoT platforms and edge computing nodes can also be included in this layer.
- **Application layer**: It contains software applications that use the data processed and collected in the perception layer to perform tasks or obtain information through advanced analysis. This is the interface through which the end user connects directly.

1.2 IoT security

"The 'S' in IoT stands for security" is a phrase commonly used when designing systems incorporating IoT devices. This phrase reflects the reality that some IoT systems lack robust security, due to a range of factors [2]:

- **Remote exposure**: Unlike other technologies, IoT devices have a large attack surface due to their internet-backed connectivity and the fact that they are typically located in an open space, giving hackers the ability to interact with devices remotely and on-site. This context helps to make attacks in IoT more effective than attacks on servers backed up behind firewalls.
- **Lack of operating systems (OS)**: IoT devices tend not to have operating systems, but rather firmware. As a result, any security issues in this are easily accessible to an attacker.
- **Manufacturers' lack of interest in security**.
- **Resource constraints**: not all IoT devices have the computing power to integrate firewalls.

As a result, IoT devices are vulnerable to large-scale attacks such as denial of service (DoS), man-in-the-middle (MiTM) attacks, malware infection, spyware, etc. These vulnerabilities can have repercussions across the entire internet, as a hacked IoT object can be used for distributed denial of service (DDoS). According to the Nokia Threat Intelligence report, IoT botnet DDoS attacks increased five times between 2022 and 2023 due to the war in Ukraine. By 2023, 40% of all DDoS traffic will come from IoT botnets [3].

1.3 Fundamentals of IoT security

In this section, we define the standard security principles with the definitions that our system aims to provide [4]:

- **Confidentiality**: The message cannot be read by anyone other than the sender and the recipient (legitimate persons).
- **Availability**: This service ensures that the system is always accessible to legitimate users.
- **Integrity**: This service ensures that messages cannot be altered during transmission between sender and receiver.
- **Authentication**: This guarantees that the message is sent to the intended person and received by the right person.
- **Non-repudiation**: A sender cannot deny having sent a message or a receiver cannot deny having received the message.

- **Authorization**: This service controls what an authenticated user is allowed to do.
- **Anonymity**: The sender and receiver cannot be identified.
- **Revocability**: This service provides the ability to revoke the access rights of any user.

1.4 Access control mechanisms

Access control refers to the mechanisms used to restrict access to resources (devices, data, applications) to authorized users or devices only, according to specific rules.

In an Industrial Internet of Things (IIoT) environment, this means managing who or what can interact with which industrial equipment, at what time, and under what conditions. In the following, we present the main types of access control.

1.4.1 Discretionary access control (DAC)

Discretionary access control (DAC) [5] is a security model in which the owner of a resource (such as a file or folder) determines who can access it and what rights they are granted. This type of access control relies on the identity of users or their groups and allows the owner to transfer or modify access permissions. Figure 1.1 illustrates how the DAC access control protocol works.

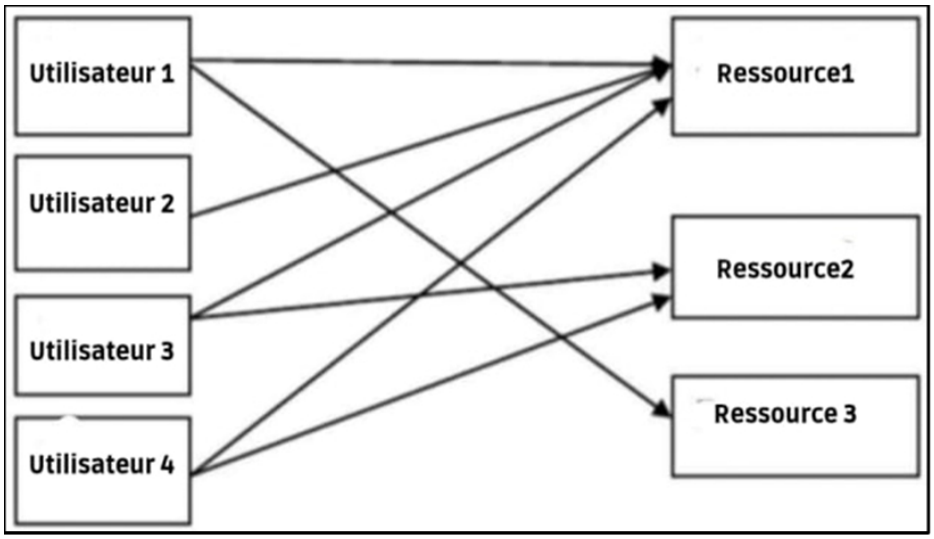


Figure 1.1: Access control model (DAC) [5]

1.4.2 Mandatory access control (MAC)

Mandatory access control (MAC) [6] is a security model in which access to resources is determined by a centralized policy, based on the classification levels of subjects (active entities requesting access to objects) and objects (passive entities storing information) in the system (e.g. confidential, secret). In this access control model, users cannot modify these rules. Figure 1.2 shows how the MAC access control model works.

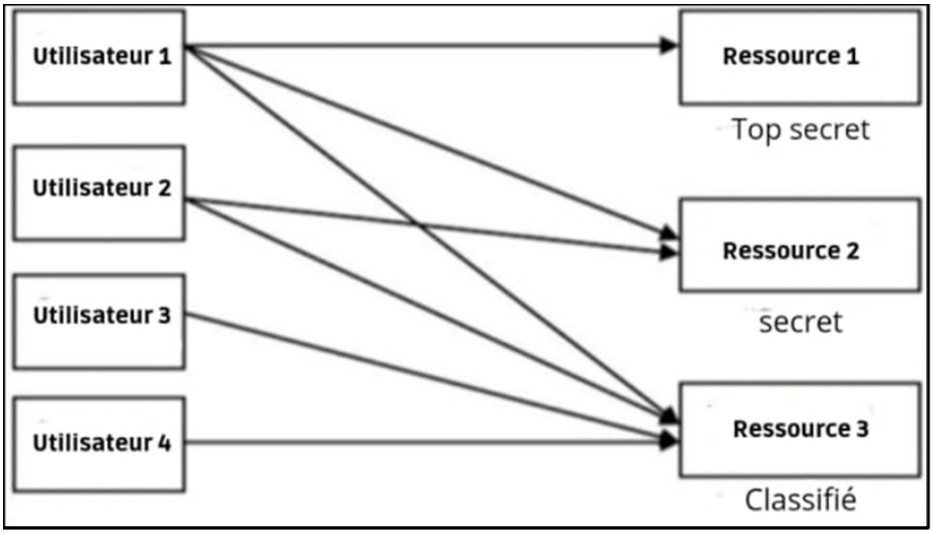


Figure 1.2: Mandatory access control (MAC) [6]

1.4.3 Role-based access control (RBAC)

Role-based access control (RBAC) [7] is a security model in which permissions are assigned to roles rather than individual users. A user is assigned one or more roles, and each role determines the actions it can perform.

The RBAC model allows centralized and simplified management of access rights, particularly in large organisations. Figure 1.3 shows how the RBAC access control model works.

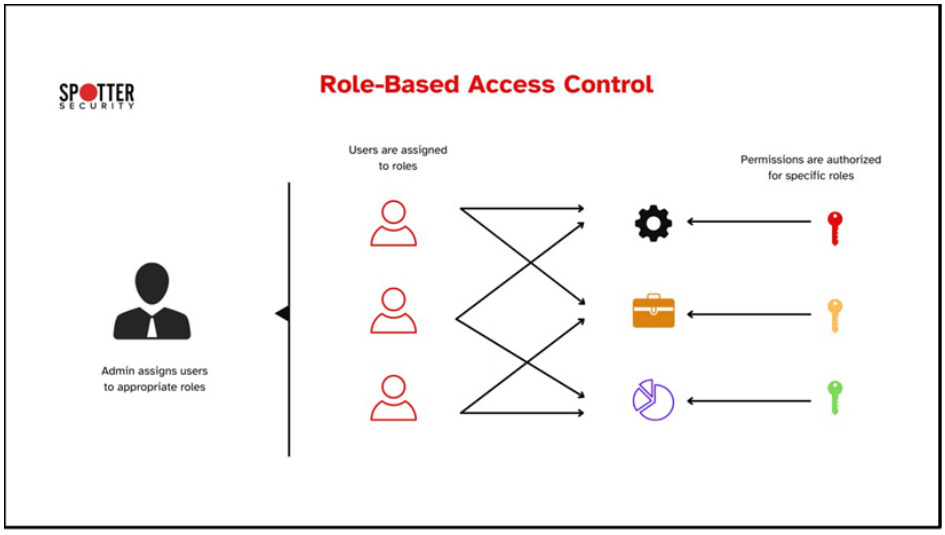


Figure 1.3: Access control model (RBAC) [7]

1.4.4 Attribute-based access control (ABAC)

The Attribute-Based Access Control (ABAC) model [8] is a model in which access decisions are made based on a set of attributes related to:

- The user (e.g. role, specialism);
- The resource (e.g. type, sensitivity level);
- The environment (e.g. time, location, etc.);
- The requested action.

Unlike the RBAC access control model, the ABAC model is not limited to roles; it uses a dynamic and precise policy, adapted to each context. Figure 1.4 illustrates how the ABAC access control protocol works.

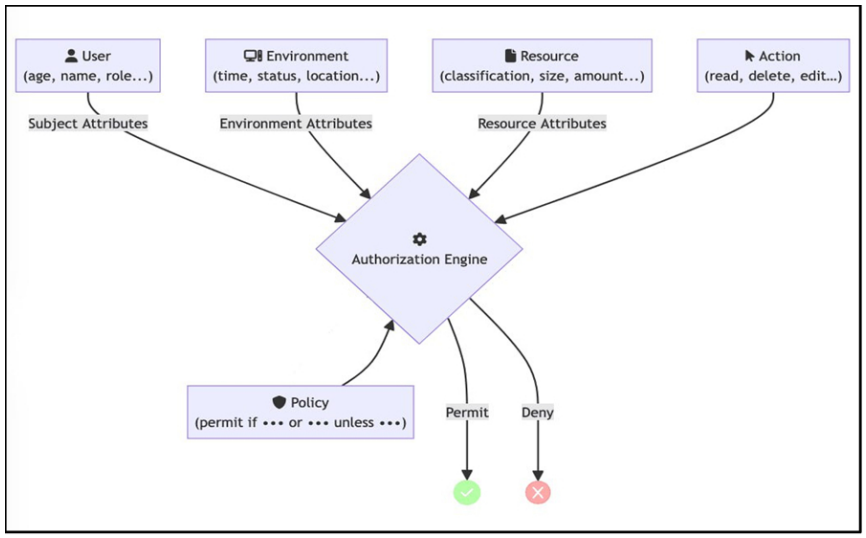


Figure 1.4: Access control model (RBAC) [8]

1.4.5 Multi-authority attribute-based access control (MA-ABE)

The MA-ABE [9] is an advanced form of attribute-based encryption "Attribute-based encryption (ABE)". Unlike traditional ABE model, where a single authority manages all attributes, MA-ABE allows multiple independent authorities to each manage a subset of attributes.

This model better meets the requirements of complex and distributed environments such as industrial IoT environments, where attributes may depend on several entities.

In our project, we used this access control model to manage access to IoT objects in the system.

1.5 Resource-limited devices

The most venerable IoT devices are those that are resource constrained, meaning they have low processing power and/or low RAM memory. This allows them to reduce costs and energy consumption, making them ideal in some areas, but on the flip side, they lack security, particularly when it comes to protect execution of sensitive tasks. One solution to this problem is

to restrict access to these IoT devices to the local network and prohibit their connection to the Internet. However, this approach can fail if a hacker infiltrates the network [10].

Table 1.1: Specifications for two widely used IoT microcontrollers and the Raspberry Pi-4 [11, 12, 13]

Device	**Processor**	**Clock speed**	**RAM**	**ROM**	**Other specifications**
ESP32	Dual/single-core 32-bit	160 or 240 MHz	520 kB	448 kB	-Ultra-low-power (ULP) co-processor -Cryptographic hardware acceleration: AES, SHA-2, RSA, ECC RNG - Wi-Fi 802.11b/g/n support - Bluetooth 4.2 and BLE support
ESP-8266	Single core 32-bit	80 or 160 MHz	160 kB (96 kB Data)	512kB-4MB	Wi-Fi 802.11 b/g/n support
Raspberry Pi 4 B	quad-core 64-bit	1.8GHz	1, 2 or 4GB	External	- Wi-Fi 802.11ac support - Bluetooth 5.0 and BLE support

The ability of these IoT objects to handle cryptographic operations is limited, so using the standard "Transport Layer Security and Secure Sockets Layer (TLS/SSL)" security protocol over Hypertext Transfer Protocol (HTTP) to secure data transfer is impractical, especially when multiple requests are unavoidable. For example for 'ESP8266', it is recommended that all TLS/SSL sketches operate at 160 MHz and not 80 Mhz. Furthermore, even at 160 MHz, some key exchanges can take several seconds to execute [14].

1.6 Attack vectors on IoT devices

Attacks on IoT systems can take several forms, as shown in Figure 1.5. In the following, we detail each type of attack on IoT objects.

1. **Physical attacks**: The attacker manipulates the hardware component using expensive materials. One example is micro-probing, which allows the attacker to read and/or inject data into the chip of IoT objects. It is used to modify the firmware or obtain confidential data about the device.

2. **Side Channel attacks**: If a device has secure communication, the attacker can retrieve information about the encryption by analyzing characteristics such as the timing, energy consumption and electromagnetic emissions of the IoT object.

3. **Cryptanalysis attacks**: The attacker attempts to break the communications encryption key using various methods such as brute force attacks, dictionary attacks, MiTM attacks, etc.

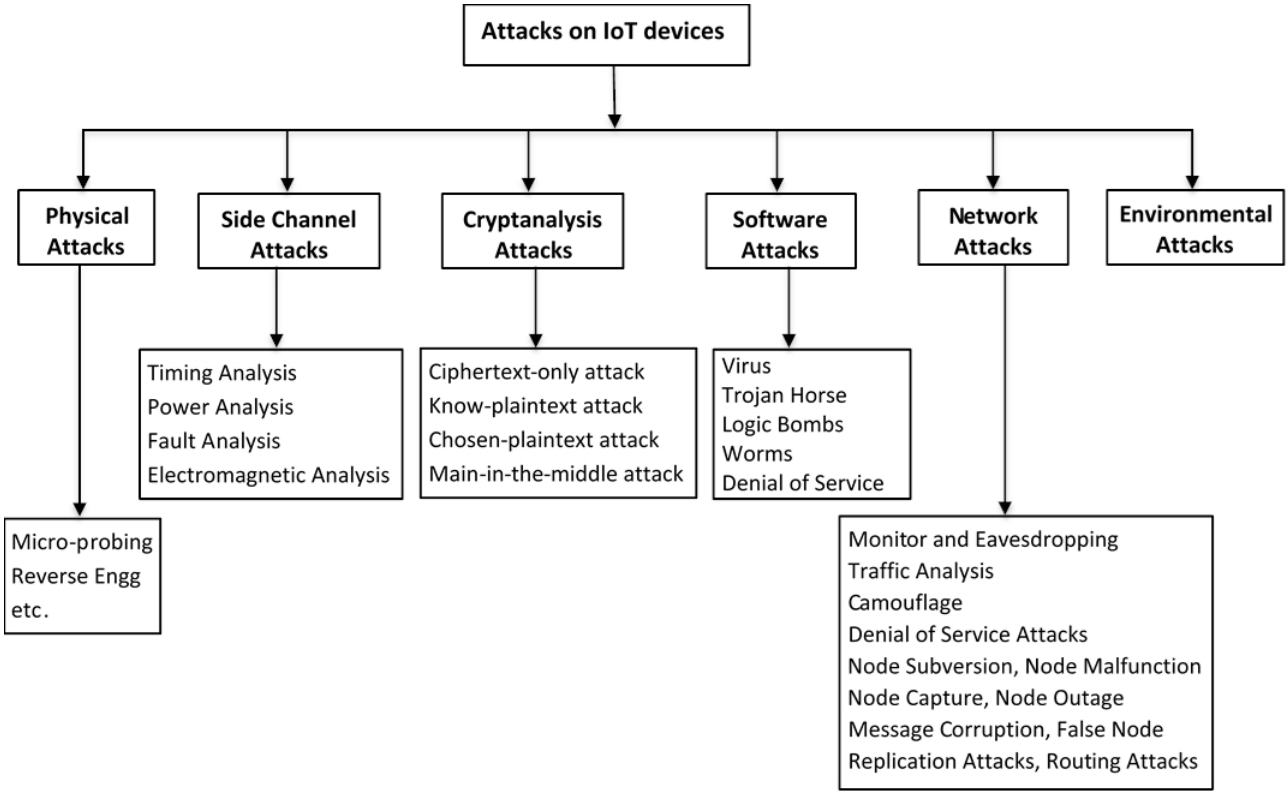


Figure 1.5: Attack vectors on IoT devices

4. **Software Attacks**: The attacker exploits system vulnerabilities to deploy malicious code. A very common vulnerability is the buffer overflow, which occurs as a result of coding errors and mistakes in software development. It is more common in languages that do not incorporate memory security, such as C and C++.

5. **Network Attacks**: The attacker uses the broadcasting of wireless communications to carry out passive and/or active attacks [15].

1.7 Computing cost of encryption

Encryption is a process carried out by the processing unit to protect data. This operation has a cost, usually measured in terms of time. The computational cost of encryption depends on several factors, such as the type of encryption, the size of the key, and the size of the encrypted data [16]. Symmetric encryption has a lower computational cost and is suitable even for IoT devices with limited resources, whereas asymmetric encryption is computationally expensive, with the cost varying according to the algorithm used. The size of the data also influences the cost of encryption. Symmetric encryption tends to scale linearly with data size, while asymmetric encryption scales exponentially. A cryptographic accelerator can be used to offload the computation onto a dedicated coprocessor, but this simply saves time and takes the work off the CPU.

1.8 IoT communication protocols

In this section, we present the main communication protocols used in IoT environments, highlighting their characteristics, strengths and limitations.

1.8.1 IoT communication over http

HTTP is an application layer protocol for transmitting multimedia data using the client-server model. Best practice is to use HTTP in combination with TLS (HTTPS) for devices that are not limited in terms of processing power and memory. For limited devices and in many cases, HTTP is not suitable because of these limitations:

- HTTP is a heavyweight protocol designed to enable reliability. By using the "Transmission Control Protocol (TCP)" transport layer protocol, it ensures reliability, but this comes with trade-offs: the excessive number of messages generated by TCP can waste bandwidth, processing time and memory that constrained IoT devices already lack.
- The HTTP protocol uses the request-response model, which limits communication to one direction at a time. It also slows down the system by using synchronous messages. IoT devices, which have limited computing resources, cannot work efficiently in synchrony and are designed for event-based communication.
- HTTP does not support multicasting, which is essential for managing multiple IoT devices [17].

In general, the use of any TCP-based protocol is not recommended when working with constrained IoT devices, as mentioned above. User Datagram Protocol (UDP) is best suited to this situation as it reduces the number and size of packets [18]. To secure UDP communications, Datagram Transport Layer Security is an implementation of TLS over UDP that protects privacy, integrity and authentication.

1.8.2 MQTT protocol

Message Queuing Telemetry Transport (MQTT) is a transport protocol designed for constrained devices and real-time data communication. It uses the client/server + publish/subscribe model and operates on the TCP protocol [19]. MQTT communication can be secured via "Transport Layer Security (TLS)" and "Datagram Transport Layer Security (DTLS)".

1.8.3 CoAP protocol

The Constrained Application Protocol (CoAP), as defined in RFC 7252, is a transfer protocol designed for constrained IoT objects. CoAP shares similarities with HTTP, such as the use of RESTful services and a client-server architecture, but unlike HTTP, CoAP uses UDP as the

transport layer. It can also support the publish/subscribe model through the OBSERVE option, which allows clients to receive updates when the state of a resource changes.

There are also other specifications for CoAP, such as CoAP over TCP (RFC 8323), Bluetooth Low Energy (BLE), SMS and even over HTTP (RFC 8075) [20]. A comparison between the COAP protocol and the HTTP protocol is shown in Table 1.2.

Table 1.2: Comparison between CoAP and HTTP

Features	**COAP**	**HTTP**
Reliability	By CoAP it self	By TCP
Caching	Yes	Yes
Multicast	Yes	No
Model	Client-Server, also supports subscribe/publish model	Client-Server
Fragmentation of large messages	Using block wise (RFC 7959)	By the physical layer
Purpose	For constrained devices	General purpose

To secure CoAP communication, DTLS can be used in several modes: Pre-Shared Key (PSK), Public Key, or using an X.509 certificate. Another security option is Object Security for Restricted RESTful Environments (OSCoRE), which is a protocol specifically designed to be integrated with CoAP. OSCoRE, as defined in RFC 8613, operates at the application layer, below the CoAP layer. It transforms the original CoAP packet into an OSCORE packet by adding an OSCORE option, which enables End-to-End Encryption (E2E) [21]. It also requires less computing power than DTLS, making it more suitable for constrained environments such as IoT.

A comparison between the OSCoRE protocol and the DTLS protocol is shown in Table 1.3.

OSCoRE requires a security context to be established beforehand. This context consists of an Authenticated Encryption with Associated Data (AEAD) algorithm, an HMAC-based Key Derivation Function (HKDF), a secret key and other parameters. This is done either manually by defining a shared context, or by using the Ephemeral Diffie-Hellman Over COSE (EDHOC) protocol, which establishes a common security context between two devices over an insecure channel.

This platform jointly uses EDHOC, OSCoRE and CoAP to enable secure communication with the IoT devices in the system.

Table 1.3: Comparison between OSCoRE and DTLS

	DTLS	**OSCoRE**
Security Layer	Transport Layer	Application Layer
Encryption Type	Hop-to-hop encryption	End-to-end encryption
Setup	Session setup required (handshake)	Common security context required
Transport layer	Over UDP	Can be run over UDP,TCP,BLE and SMS
Multicast support	No)	Yes
Handshake	Built in internally	relies on pre-established context
Use Cases	Secure sessions in semi-constrained environments	E2E security in constrained networks with proxies and multicast

1.9 Conclusion

In this chapter, we analyzed the security of IoT devices, focusing on their specific vulnerabilities and the most dangerous attack vectors. We have also explored the main access control mechanisms described in the literature. Then, we examined the unique challenges associated with securing communication to resource-constrained IoT devices and explored approaches for establishing it.

The next chapter is describing the platform , the actors roles and general conception of the system.

CHAPTER 2

ARCHITECTURE OF THE PROPOSED PLATFORM

2.1 Introduction

This chapter focuses on the requirements specification of our system and its design. We began this chapter by enumerating the purpose of the platform, followed by an explanation of some of the design choices we use in the platform that are necessary to understand the design of the system.

2.2 Objectives of the platform

The platform developed tackles three main objectives:

- Secure data transfer between IoT objects and end users. It must use fewer computations that can be performed by constrained IoT devices while providing the necessary security.
- Manage user access control to IoT objects. Although this can be achieved relatively with a single responsible authority, we add the following caveat: multiple authorities can co-exist in the system, each managing their own access control to their devices. This means that if a device is under the responsibility of several authorities, the end user must obtain authorization from all of these authorities at the same time to be able to access the device. This is achieved using MA-ABE.
- The use of a decentralized architecture with the inclusion of fog computing. The inclusion of fog computing is necessary for an industrial environment in order to reduce latency between IoT devices and end-users.

2.3 MA-ABE and access control

In this section, we present in more detail the approach used to develop our platform and how access control to objects by end users is managed in our system.

2.3.1 Identity-based encryption (IBE)

Identity-based encryption (IBE) is an encryption system based on public key cryptography. It allows an identity (for example, an email address) to be used as the public key, eliminating the need for a public key infrastructure (PKI) to protect against MiTM attacks [22].

2.3.2 Attribute-based encryption (ABE)

ABE is an encryption system developed from the IBE cryptographic system. It allows an authority (which can be an entity or a person) to distribute attributes to users of the system as it sees fit. Users can encrypt messages using the attributes given by the authority, and only users with the necessary attributes can decrypt the message [23].

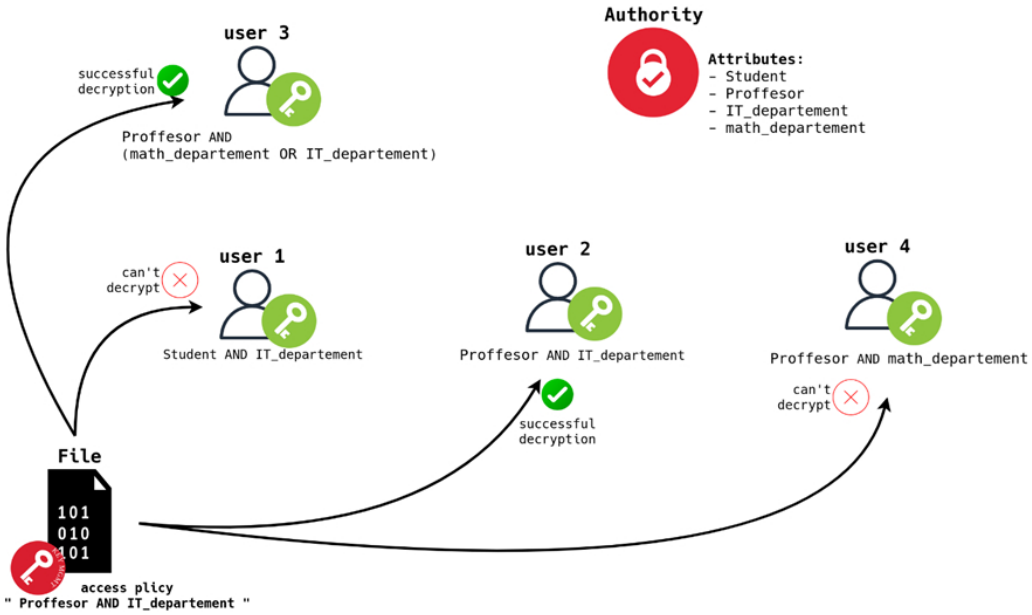


Figure 2.1: Illustration of the ABE approach

For example (Figure 2.1), a university, which is an authority, manages these attributes [Student, Professor, Computer science department, Mathematics department]. These attributes are allocated to users according to their role and the department to which they are assigned. Now let's suppose that a Professor of the computer science department wants to send a confidential exam file to his colleagues. He encrypts it using the following access policy: (Professor AND Computer science department). This means that only users with both attributes (a professor from the computer science department) will be able to decrypt and read the file. A professor from the maths department cannot decrypt the file, nor can a student from the computer science

department. This allows the encrypted file to be stored publicly. Authorization is provided by the ABE.

The ABE approach has two architectures:

- **Cyphertext policy attribute based encryption (CP-ABE)**: the access policy is specified during encryption, which is then transmitted with the message. Decryption requires the user's keys to satisfy the access policy.
- **Key Policy Attribute Based Encryption (KP-ABE)**: the access policy is specified when the user's keys are distributed. During encryption, the user specifies the attributes and only users with an access policy with the correct attributes can decrypt the message [24].

Figure 2.2 illustrates the difference between the CP-ABE architecture and the KP-ABE architecture.

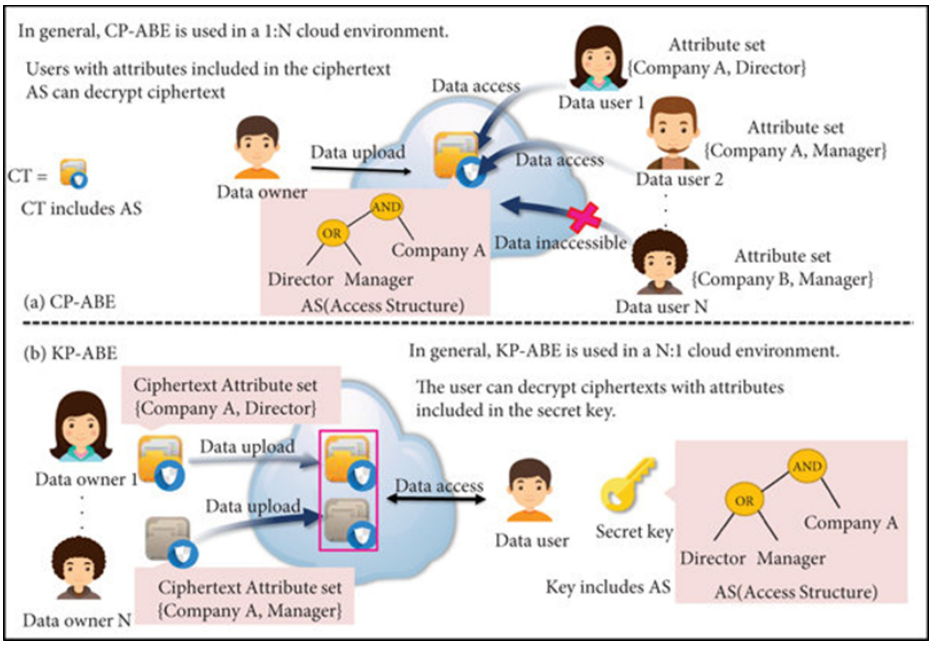


Figure 2.2: Difference between CP-ABE and KP-ABE

2.3.3 Multi Authority Attribute Based Encryption (MA-ABE)

MA-ABE is a decentralized encryption scheme based on CP-ABE. It allows for the existence of multiple independent authorities (any party can be an authority). Each can distribute its own set of attributes. Users can encrypt using multiple attributes from multiple authorities, and only those who satisfy the message's access policy can decrypt it [9].

In the previous example, if we divide the system into several authorities (the university's departments). Each department assigns its own attributes to system users:

- Computer Science Department Issues:

$$[Professor_{Dep\_Computer\_Science}, Student_{Dep\_Computer\_Science}, AssistantTP_{Dep\_Computer\_Science}]$$

- Math Department Issues:

$$[Professor_{Dep\_Maths}, Student_{Dep\_Maths}]$$

A professor in the computer science department wants to send a confidential document to his colleagues in the mathematics and computer science departments, so he encrypts it using MA-ABE with the following access policy:

$$(Professor_{Dep\_Computer\_Science} \quad OR \quad Professor_{Dep\_Maths})$$

Only users with one of the two attributes can decrypt the message. The Figure 2.3 illustrates an example of MA-ABE approach.

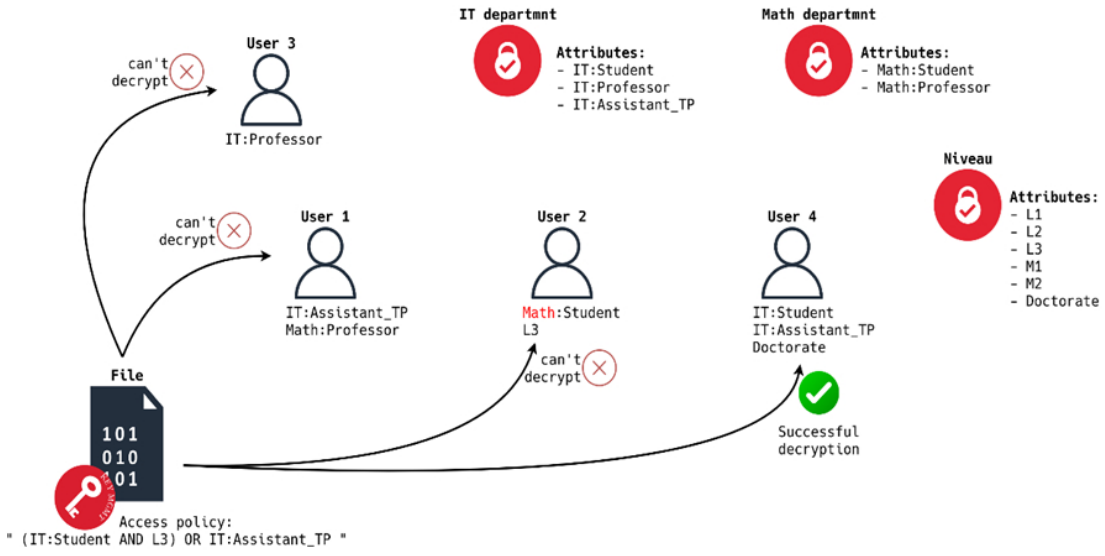


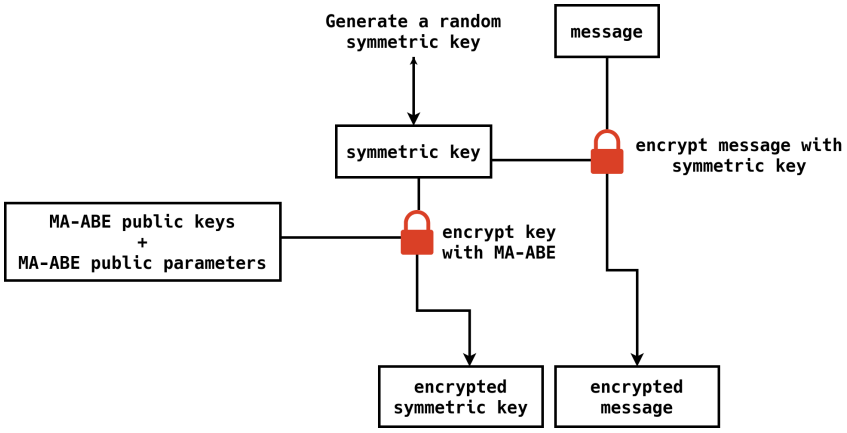
Figure 2.3: Example of Multi-Authority ABE

MA-ABE is immune to collusion [1]: for example, if we have authorities A and B with attributes $[a_1, a_2, a_3]$ and $[b_1, b_2, b_3]$ respectively and users defined as $(user_1[b_1, a_2])$ and $(user_2[a_1])$. A message encrypted using "$(a_1\ AND\ (b_1\ OR\ b_2))$" cannot be decrypted by combining the keys of two users, even if they collectively have the required attribute.

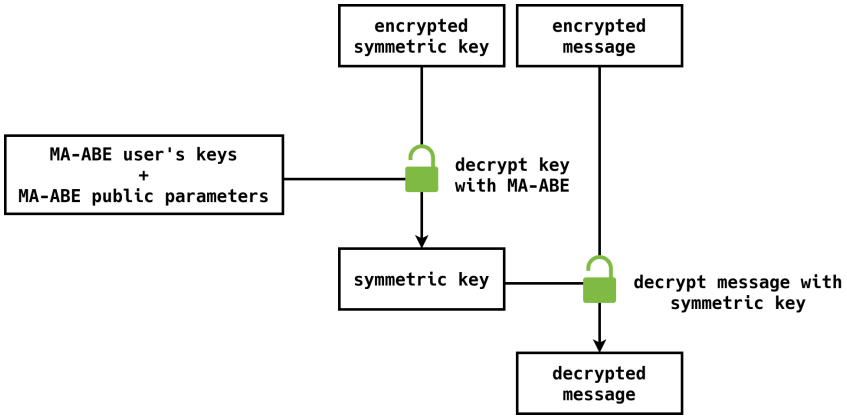
The MA-ABE model is costly in terms of computation and energy consumption, especially for equipment with limited capacity, so encrypting and decrypting large messages is not viable. It is therefore combined with symmetric encryption to improve performance. The idea is to encrypt the message with a randomly generated symmetric key of fixed length (such as AES-

[1]Several users do not have the required attributes individually, so they combine their keys in an attempt to decrypt an encrypted text

256), then encrypt this key with MA-ABE, including the encrypted key in the ciphertext. The decryption process is the reverse of the encryption operation. The figures 2.4b and 2.4a illustrate the encryption and decryption processes respectively in the MA-ABE model.



(a) MA-ABE Encryption process



(b) MA-ABE Decryption process

Figure 2.4: Encryption and decryption in MA-ABE

A system that uses MA-ABE encryption is composed as follows:

- Public parameter: this is generated once during the configuration phase. It is accessible to all parts of the system and is used for all operations.
- Authority: entity that manages the attributes and holds the database of users and their attributes. Each attribute is associated with a public-private key pair. The public key is used for encryption operations, while the private key is used to generate the user's key for that attribute.
- User: for each attribute that the user has, the key is retrieved from the authority and can be used to decrypt a message if it contains the attribute in the access policy.
- Cyphertext: the message is encrypted using the public parameter, the access policy and the public keys of all the authorities. It can be decrypted using the user's keys if their keys satisfy the access policy.

In the platform developed, MA-ABE is used to control user access to IoT devices by several authorities independently of each other.

2.4 Integration of fog computing into the platform

Fog computing involves offloading some of the computations from the centralized node (cloud) to smaller nodes located close to edge devices (fog being closer to the ground than clouds are). This allows computations to be distributed and enables low-latency real-time applications.

It is also advantageous to have a distributed architecture, where even if one node fails, another can replace it. This contrasts with a centralized architecture where, if the central node fails, users cannot access IoT data i.e. IoT devices [25]. The Figure 2.5 illustrates the integration of the fog computing in the platform developed.

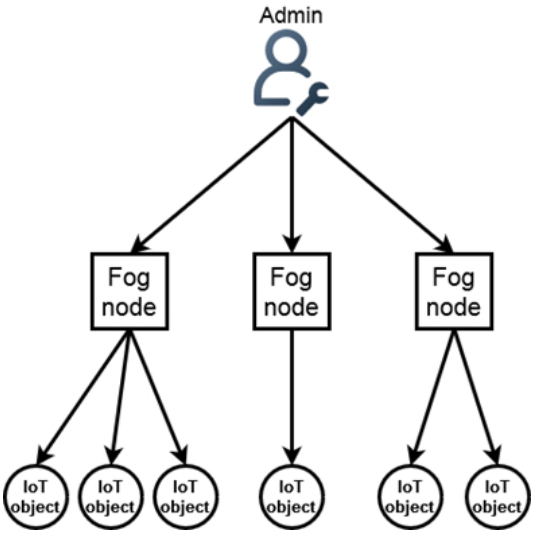


Figure 2.5: Incorporation fog computing in the platform

2.5 Specification of system requirements

2.5.1 Identification of actors

- Personnel (user): these are the system personnel who operate on the IoT devices (monitoring and controlling them) from a mobile device that is not computationally or memory-constrained. They require appropriate authorization to access these devices.
- Authority: is the entity within the organization that is responsible for giving users their roles in the system.
- Admin: is the manager of the system. It is responsible for controlling access to all IoT devices in the system and managing both users and authorities.

2.5.2 Functional requirements

1. **Admin**

   - **User management**: the administrator can create, update and delete users who can access the system; he can also define their authentication passwords.
   - **Authorities management**: the administrator can view, accept and reject registration requests. They can also delete authorities that have already been registered.
   - **List fog nodes**: the administrator can view fog nodes.
   - **IoT device management**: the administrator can add, update and delete IoT devices, assign them to specific fog nodes and define a device access policy.

2. **Authority**

   - **Managing roles within the organization**: the authority can consult, create, update and delete roles. These roles define user authorizations.
   - **Manage users**: the authority can view, create and delete the users it employs. It can also assign and delete roles to users.
   - **Request to be registered**: the authority can request to be registered in the system.

3. **Users**

   - **Retrieve access authorization from authorities**: the user can retrieve proof of their role within the organization and store it.
   - **List IoT devices**: the user can view devices in the system and see their current status.
   - **Request access to devices**: the user can request access to devices.
   - **Access IoT devices**: the user can operate on IoT devices by sending commands and monitoring sensor outputs in real time.

2.5.3 Non-functional requirements

- **Confidentiality and security**: all communications in the system must be secure against active and passive attacks.
- **Constrained environment**: the system must take into account resource-constrained devices and be designed to accommodate their limitations.
- **Availability**: All devices must be available at all times.
- **Authorization**: Only authorized users can access their specific IoT devices. Access control must be robust.

2.6 The design of the system

In this section, we present the design of the proposed platform, using the use case diagram and sequence diagram to illustrate how it works.

2.6.1 User case diagram

We illustrate the external entities (actors) that interact with the system, the functionality offered by the developed system and how actors and use cases connect (relationships) using a use case diagram as shown in Figure 2.6.

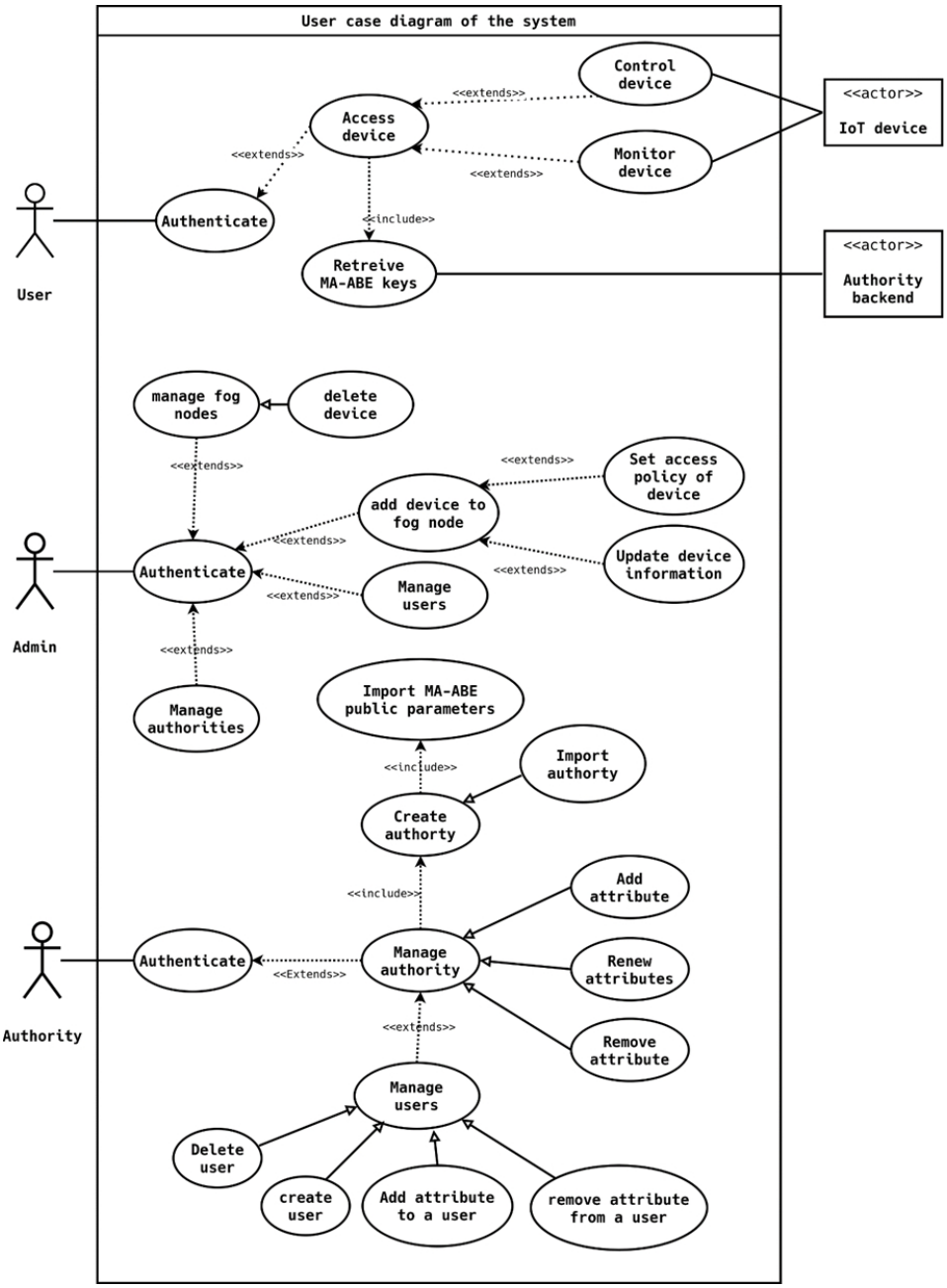


Figure 2.6: User case diagram of the platform proposed

2.6.2 Sequence diagram

We use a sequence diagram to show how objects/components interact in the system over time, while focusing on the order of messages exchanged to achieve a specific scenario.

2.6.2.1 Various authority scenarios

1. **Establishment of a new authority**
   Setting up a new authority involves the following process and is shown in figure 2.7:

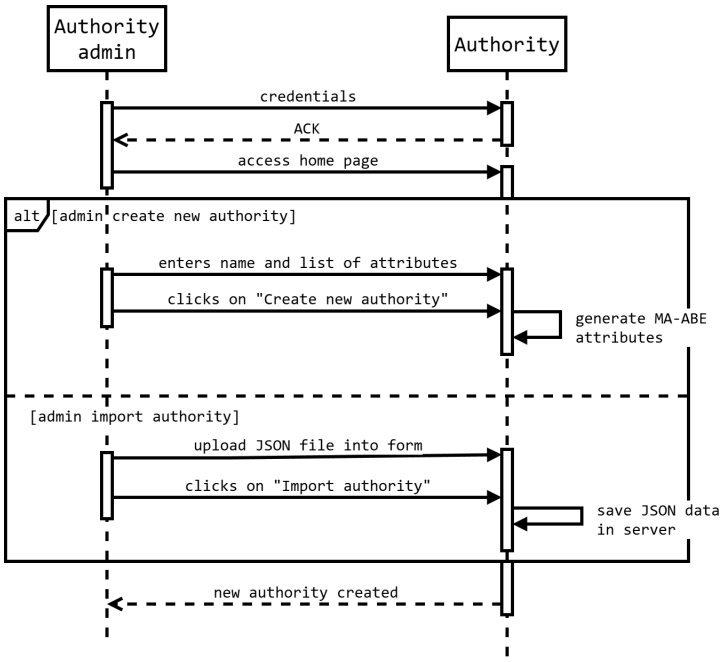


Figure 2.7: Sequence diagram for Establishment of a new authority

   (a) The admin authenticates and accesses the home page for the first time.

   (b) The admin chooses between creating a new authority or importing a pre-existing authority.

   (c) Select *"create a new authority"* and enter the name of the authority.

   (d) Enter the list of attributes that the authority will manage.

   (e) The admin clicks on *"Create an authority"*. This creates the MA-ABE authority and, for each attribute, generates the corresponding public and private keys.

   The admin can also choose to import a pre-existing authority:

   (a) In step 1, he chooses to import an authority.

   (b) Select a JSON file containing the authority data and click on *"Import authority"*.

   (c) If successful, the admin proceeds to step six.

If the admin selects an incorrect file or if the data cannot be read, an error message is displayed and the process is cancelled.

2. **Adding new attribute**
   To add a new attribute, the process is as follows and is shown in figure 2.8:

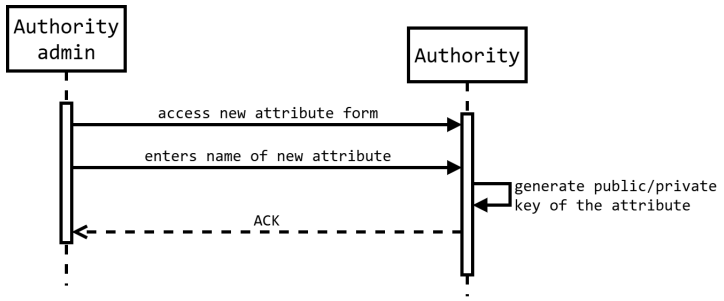


Figure 2.8: Sequence diagram for Adding new attribute

   (a) The manager of the authority accesses the web application ⟶ attributes page ⟶ form of new attributes.

   (b) Enter the name of the attribute and click on *"Add an attribute"*.

   If the manager of the authority enters an attribute that already exists, an error message is displayed and the process is cancelled.

3. **Renewing attribute keys**
   This scenario involves revoking access for a group of users with certain attributes. The revocation process is as follows and is shown in figure 2.9:

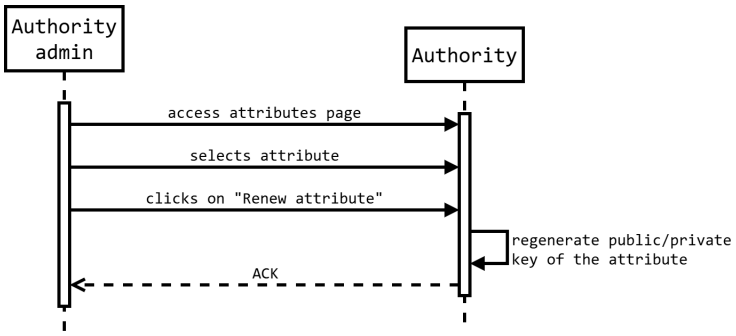


Figure 2.9: Sequence diagram for Renewing attribute keys

   (a) The manager of the authority enters the web application ⟶ attributes page ⟶ attributes list ⟶ desired attribute.

   (b) He clicks on *"Renew attribute"* and confirms the action.

   (c) The attribute's private and public keys are regenerated.

(d) The system notifies all fog nodes that manage objects whose access policies include the renewed attribute. These objects perform the registration process again (see the IoT object registration scenario).

4. **Adding a new user**

The process for adding a new user is as follows and is shown in figure 2.10:

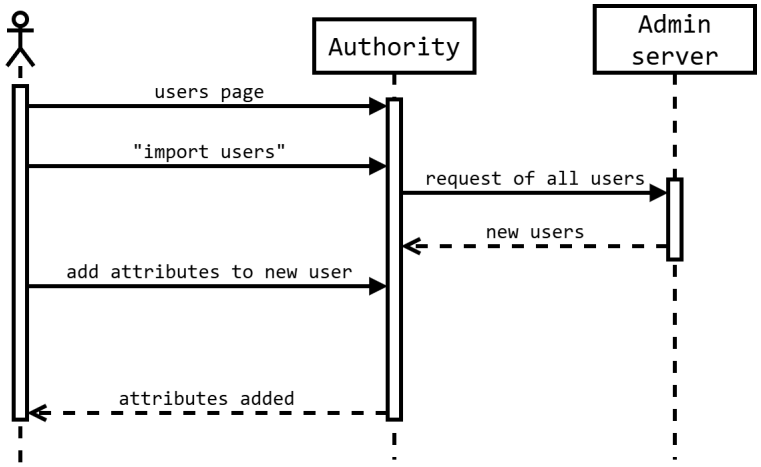


Figure 2.10: Sequence diagram for Adding a new user

(a) The manager of the authority enters the web application ⟶ users page ⟶ import users form.

(b) He clicks on *"import users from admin"*.

(c) He selects the user then selects the attributes to be assigned to him. He then clicks on *"Add attributes"*.

(d) The new user is created.

If the admin enters a name that already exists, an error message is displayed and the process is cancelled.

5. **Adding attribute to a user**

This operation is performed as follows:

(a) The manager of the authority enters the web application ⟶ user page ⟶ user list ⟶ desired user

(b) He clicks on *"Add an attribute"*. A menu of available attributes is displayed. Select an attribute and click on 'add attribute'.

(c) The selected attribute is assigned to the user.

6. **Removing an attribute from a user**

This operation is carried out as follows:

(a) The manager of the authority enters the web application ⟶ user page ⟶ user list ⟶ desired user.

(b) He clicks on *"Delete attribute"*. An attribute selection menu appears. Select an attribute and click on 'delete attribute'.

(c) The selected attribute is deleted from the user.

If the user does not yet have an attribute, the *"Delete attribute"* button is not displayed.

7. **Removing a user**
   The process for removing a user is as follows:

   (a) The manager of the authority enters the web application ⟶ user page ⟶ user list ⟶ desired user.

   (b) The manager of the authority clicks on *"Remove user"*.

   (c) The user is removed.

   The removed user will no longer be able to connect to their application.

2.6.2.2 Admin scenarios

The admin can perform several operations:

1. **Managing authorities**
   This scenario is linked to an authority sending its public keys. The keys received are in the *"pending"* section. He must be approved/refused by the central node administrator.

   (a) The admin enters the web application ⟶ authorities page ⟶ list of pending authorities ⟶ desired authority.

   (b) The admin Clicks on *"Accept authority"*.

   (c) The authority changes to the accepted authority.

   If the authority is rejected, it is removed from the system.

2. **Adding IoT devices**
   Adding a device does not mean that the device has entered the network. It will do so during the registration process.

   The IoT device is defined as being under the responsibility of a fog node. It must be on the same local network: any inconsistency will cause the device registration to fail (see the IoT device registration scenario).

   (a) The admin enters the web application ⟶ fog nodes page ⟶ desired fog node.

   (b) The admin enters the name and description of the IoT device.

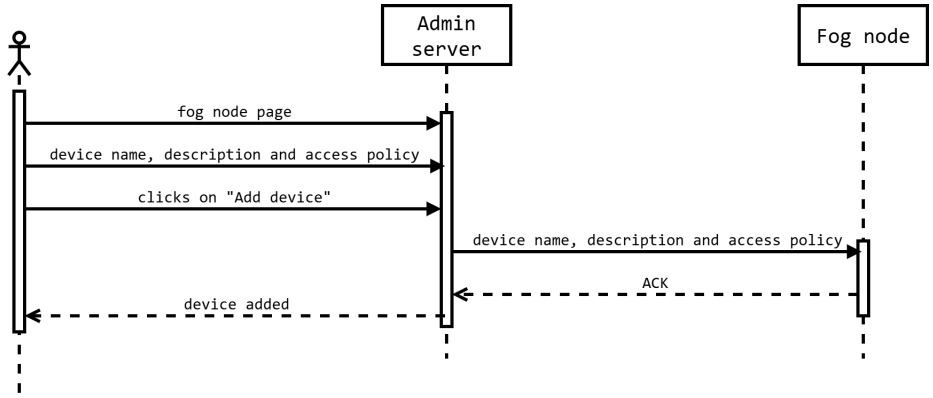


Figure 2.11: Sequence diagram for Adding IoT devices

(c) The admin enters the access policy using the accepted authority attributes. Then, he clicks on *"Add device"*.

(d) The IoT device is added and the admin sends the device data to the fog node.

(e) The fog node receives the device information and registers the device locally.

If the authority does not exist in the system, the admin cannot add a new device and if the admin enters an invalid access policy, an error message is displayed and the process is aborted.

3. **Removing IoT device from a fog node**
   The operation to remove an IoT device from a fog node is performed as follows:

   (a) The admin enters the web application⟶ fog nodes page ⟶ fog node ⟶ desired device.

   (b) He clicks on *"Remove device"* and confirms the action.

   (c) The device is deleted by the admin and he sends a request to remove the device to the fog node.

   (d) The fog node removes the object from its local storage and sends a stop command to the object.

   (e) The object receives the commands and shuts down.

4. **Updating an IoT object**
   The process for updating an IoT object is as follows:

   (a) The admin enters the web application ⟶ fog nodes page ⟶ fog node ⟶ desired device.

   (b) The admin clicks on *"update device"* and the device information can be modified. After changing the description and access policy, he confirms the action.

(c) The device is updated and the admin sends the new device information to the fog node.

(d) The fog node receives the updated information and replaces the object information with the new information.

If the device is already registered (entered into the network) and the fog node detects that the access policy has been replaced, it performs the following steps:

(a) The fog node generates a new random token of fixed length (32 bytes) and sends it to the IoT device.

(b) The IoT device receives the token and replaces the old token with the new one.

(c) The fog node encrypts the copy of the new token with MA-ABE using the updated access policy and public keys it has and stores it locally.

The IoT object will refuse any request that includes the old token as authorization.

2.6.2.3 Fog node scenarios

The fog node is not a system actor. It executes the startup process as follows and is shown in figure 2.12:

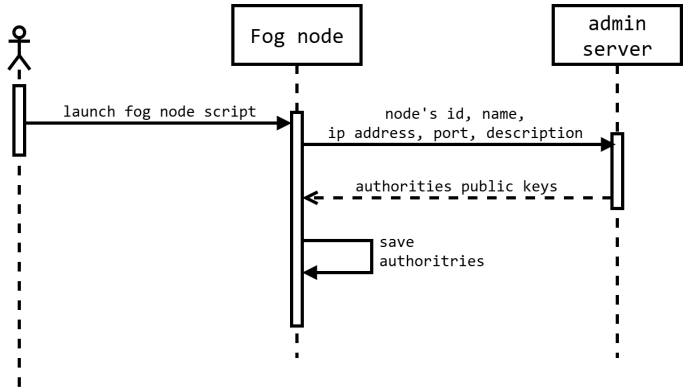


Figure 2.12: Sequence diagram for Fog node scenarios

1. The fog node's ID, node name, node description, IP address and port are entered and sent to the admin of the system.

2. The admin receives the fog node information and stores it locally. It responds with the public keys of the authorities accepted by the fog node.

3. The fog node receives the public keys and stores them locally.

4. The fog node has completed the start-up phase and is now part of the system.

If a fog node cannot locate or connect to the system, startup is aborted and the fog node automatically shuts down.

2.6.2.4 IoT device scenarios

Although an IoT object is not an actor in the system, it can perform the following actions:

1. **IoT device registration**
   An IoT device can be registered with a fog node as follows and is shown in figure 2.13:

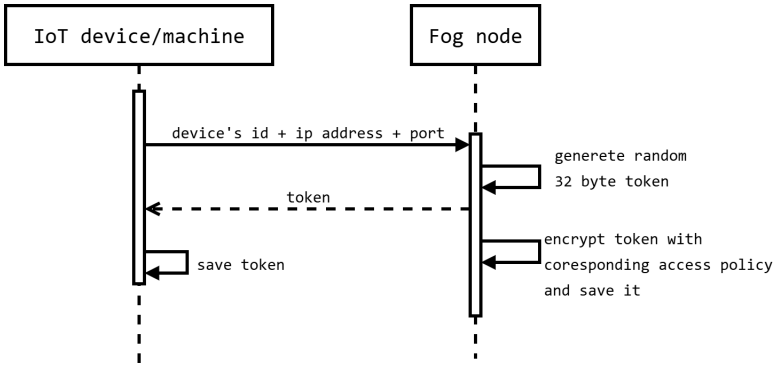


Figure 2.13: Sequence diagram for IoT device registration

   (a) The IoT device starts and sends a registration request to the fog node.

   (b) The fog node receives the request and checks if the object is defined by the admin (see Adding IoT devices scenario). If so, the fog node generates a random token of fixed length (32 bytes) and sends it back to the device.

   (c) The device receives the token and stores it locally.

   (d) The fog node encrypts a copy of the token with MA-ABE using the admin-defined access policy and public keys, then stores it locally.

   The fog node's IP address and port are preconfigured on the device via DNS and the token will be used to verify user authorization. Moreover, if the IoT device is not defined by the administrator in step 2, the fog node rejects the device request and the device shuts down.

2. **IoT object receiving a request from a user**

   The process for a user to access and operate an IoT object is as follows and is shown in figure 2.14:

   (a) The user sends a request to the object.

   (b) The object receives the request and checks whether it contains a token.

   (c) The object compares the token received with the locally stored token.

   (d) If the two tokens match, the object processes the request normally.

   If the tokens do not match or if the user does not provide a token, the device rejects the user's request with an unauthorized code.

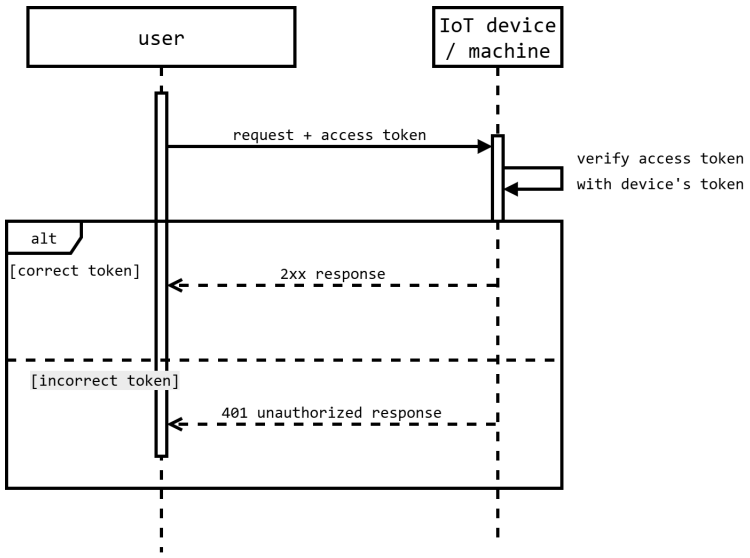


Figure 2.14: Sequence diagram for IoT object receiving a request from a user

2.6.2.5 Various user scenarios

In this sub-section we present the different actions that can be performed by a user.

1. **User retrieving MA-ABE keys**
   The process for retrieving keys by a user is based on the following algorithmic scheme:

   (a) The user enters the mobile application $\longrightarrow$ authorities page $\longrightarrow$ desired authority.

   (b) He clicks on *"Retrieve keys"*. He sends the authority a request containing his user ID.

   (c) The authority receives the request, looks up the attributes assigned to the user in the database and generates the corresponding MA-ABE user keys based on these attributes and the user's identifier. It then sends the generated keys back to the user.

   (d) The user receives the keys and stores them locally.

   If the authority cannot find the user in the database, an error message is displayed and the process is cancelled. This process is repeated for all the authorities to which the user belongs.

2. **User requesting access to an IoT device**
   The process for a user requesting access to an IoT object belonging to the system is as follows and is shown in figure 2.15:

   (a) The user enters the mobile application $\longrightarrow$ IoT objects page.

   (b) A list of devices registered in the fog node is displayed. The user selects the desired IoT device and clicks *"Access object"*. This sends a request containing the name of the object to the corresponding fog node.

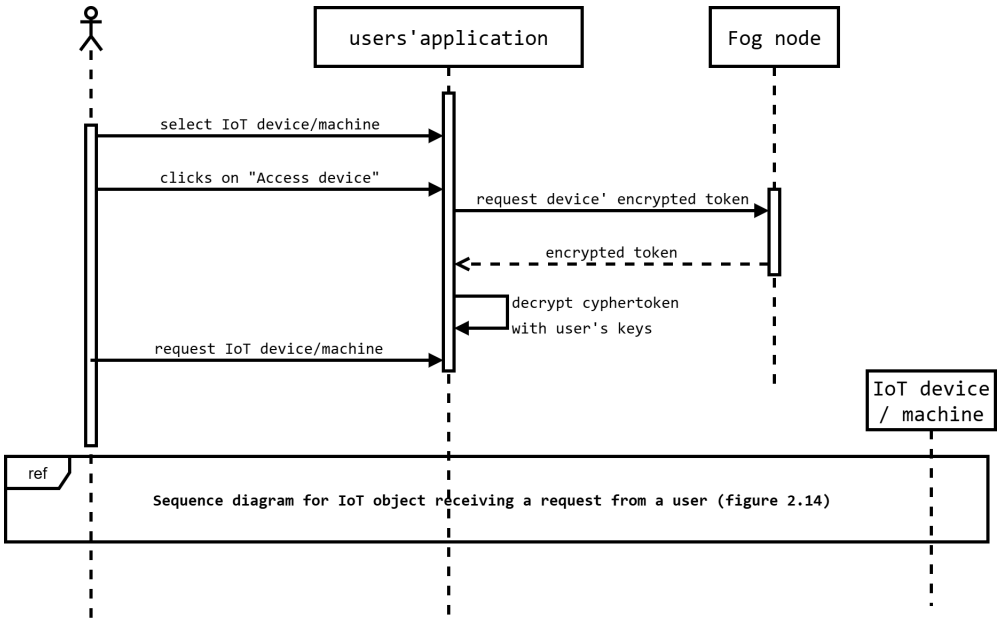


Figure 2.15: Sequence diagram for User requesting access to an IoT device

(c) The fog node receives the request, searches for the device and responds with the encrypted token for that device.

(d) The user receives the response, decrypts the encrypted token with the MA-ABE keys available to him. If successful, the *"go to object page"* button appears.

(e) He clicks on the *"go to object page"* button, and the monitoring and control page is displayed.

If the device is not entered into the network in step 2, the *"Access object"* button is disabled and a message is displayed *"object not connected"*. Moreover, if the decryption process in step 4 fails (the result of the decryption is an empty string), the following error is displayed *"You do not have permission to access the object"*. Furthermore, if the process is carried out correctly, at step 6 each object will have its own specialized page with different functionalities.

2.6.3 Class diagram

The class diagram is a visual representation used to explain system design concepts. It serves as an object-oriented blueprint, illustrating the structure of the system through its classes, attributes, methods and relationships. Figure 2.16 illustrates the structure of the proposed security system.

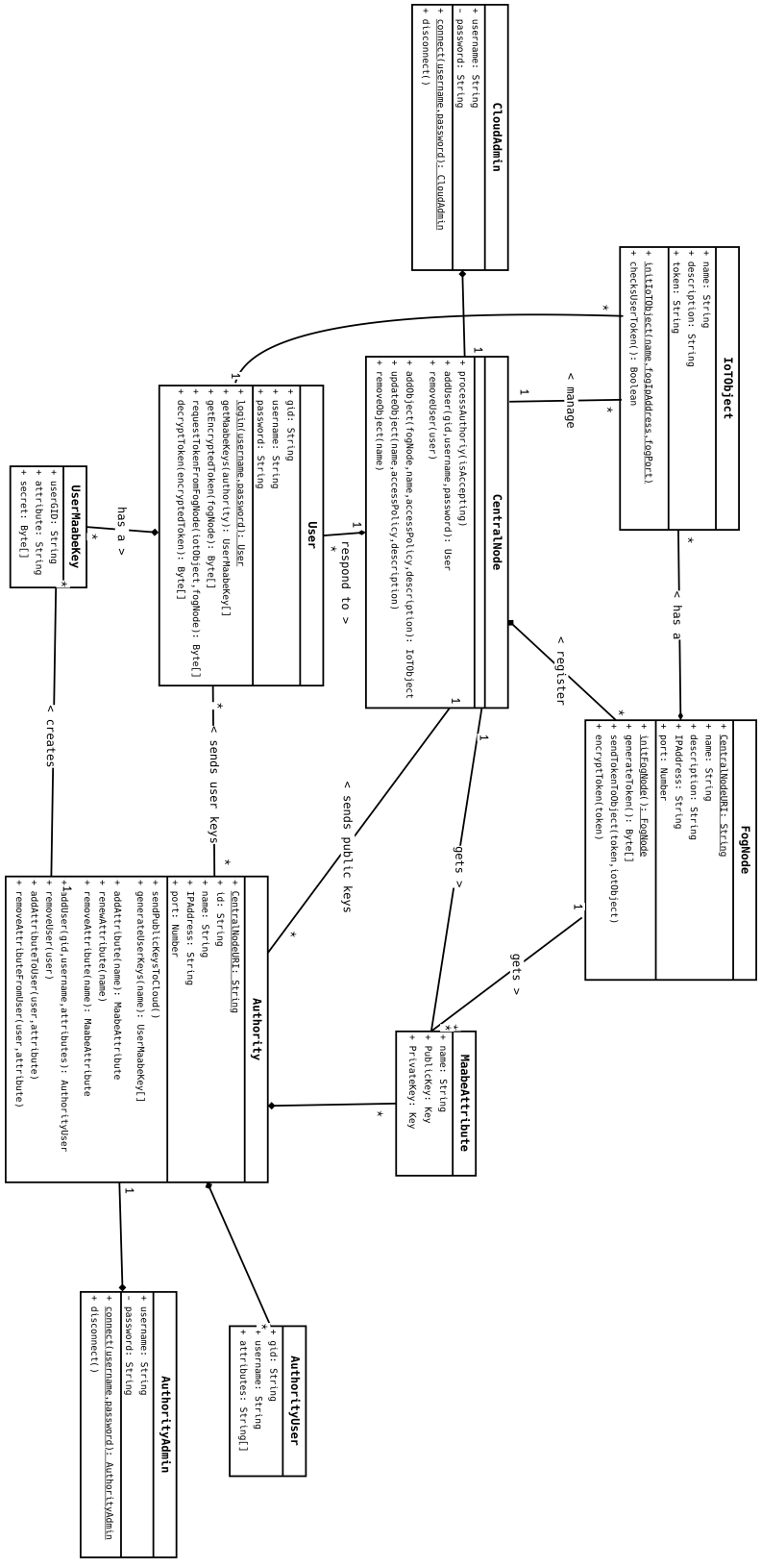


Figure 2.16: Class diagram of the proposed security system

2.7 Architecture of the proposed system

Figure 2.17 shows the architecture of the proposed security system for an IoT environment.

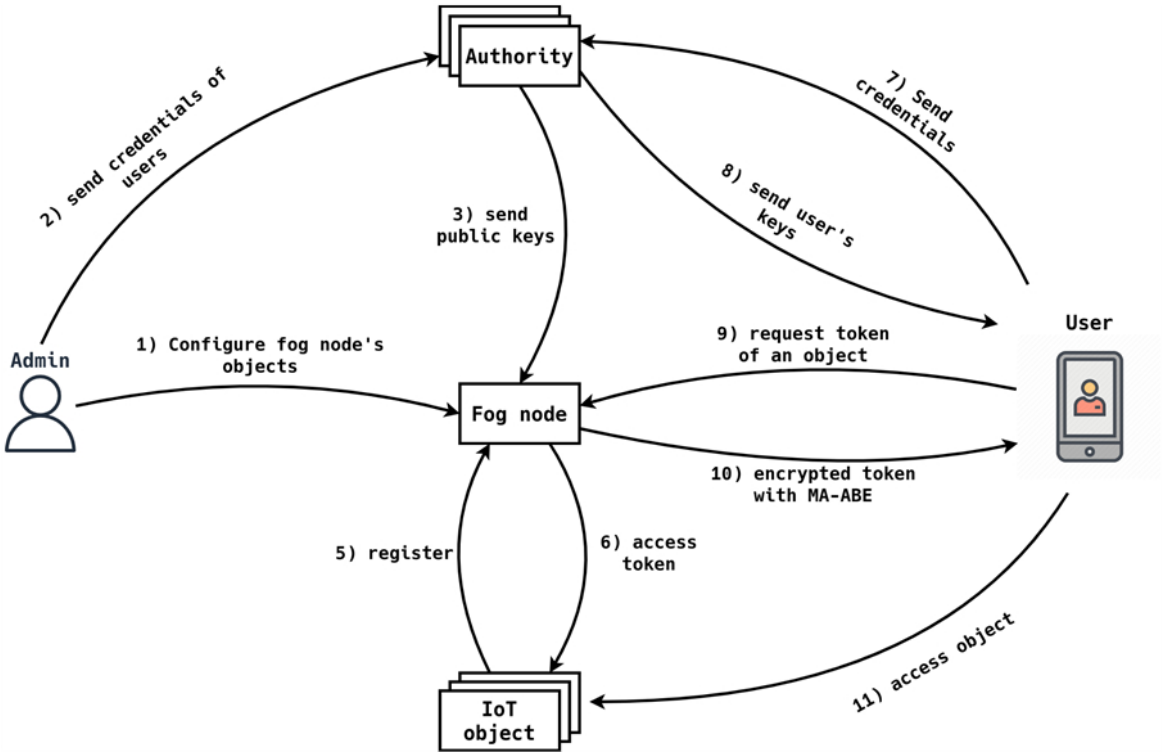


Figure 2.17: Architecture of the proposed system

The system operates in two phases (see Figure 2.18):

1. **The set-up phase** : in this phase, the following operations are performed:
   - The admin creates system users, configures fog nodes and objects.
   - The fog nodes enter the network and register with the admin server.
   - Authorities initialize their attributes and retrieve user credentials from the admin.

   This phase is run once but actions can be repeated independently at a later date.

2. **The operational phase**: in which, the following operations are performed:
   - Users retrieve their keys from the authorities.
   - IoT objects enter the network and the fog node generates their access token for them.Users request the encrypted tokens and then decrypt them to gain access.

In the set-up phase, the admin first creates the users in the system where the users need to authenticate. The user authentication process is as follows:

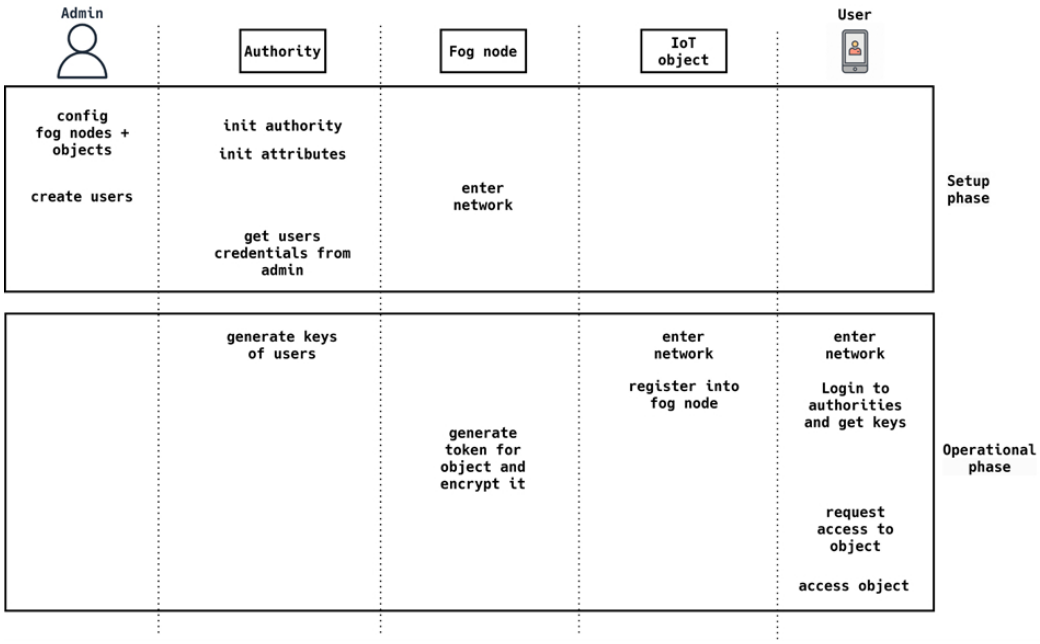


Figure 2.18: The two phases of the system: set-up and operational

- User authentication credentials must be independent for each authority: authenticating a user with several authorities will provide them with different credentials. This is to make the authorities independent of each other and to protect the user in the event of a breach or misuse of one of them.
- The user must not authenticate to each authority independently. He must authenticate once and the system must derive the credentials based on the authority.

The solution is to use the Key Derivation Function (KDF) in the scheme shown in Figure 2.19.

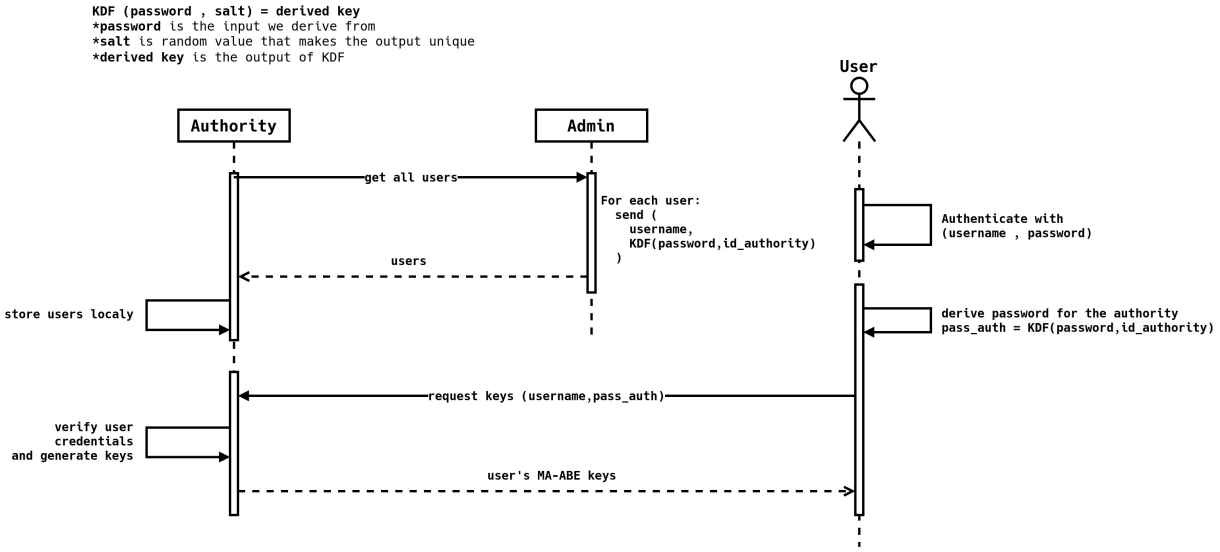


Figure 2.19: The use of the Key Derivation Function (KDF)

This authentication process solves both criteria: The authority cannot determine the $Password_{user}$ from the $key(user, authority) = KDF(Password_{user}, ID_{authority})$ and users only need one identifier to authenticate to each authority.

2.8 Conclusion

In this chapter, we have described the functionality of the system, detailing the role and actions of each entity. We have also presented the different access control models, with particular emphasis on the MA-ABE model, which is considered to be a very robust model and provides highly efficient access control.

The next chapter presents the tools used, both software and hardware, which are needed to develop this security platform for IoT environments. We also illustrate the platform through several execution examples and show the different scenarios and situations that our developed platform can handle.

CHAPTER 3

DEVELOPMENT AND IMPLEMENTATION OF AN IOT ACCESS CONTROL SYSTEM

3.1 Introduction

IoT is growing exponentially, interconnecting billions of heterogeneous devices ranging from industrial sensors to consumer-connected objects. However, with this expansion come major security challenges, particularly when it comes to controlling access to resources and data. A robust access control protocol must guarantee authentication, authorization, and confidentiality, while taking into account the specific constraints of IoT environments (limitations in computing power, memory, and energy).

In this chapter, we present a detailed analysis of the hardware and software tools required to develop and evaluate the access control protocol that we have developed. we also present through our application several scenarios on which the access control model will take place.

3.2 Conception Tools

3.2.1 Draw.io

"Draw.io" is a web-based diagramming tool used to create "UML" diagrams, architectural models and flowcharts. It offers a wide variety of templates and supports custom shape libraries, making it a flexible and powerful tool for visual modelling. In this project, *"Draw.io"* was used to design UML diagrams and to model the overall architecture of the platform developed.

3.2.2 Figma

"Figma" is a web-based design tool for creating user interfaces (UIs), wireframes [1] and prototypes. We use it to design the user application interface.

3.3 Hardware tools

To develop this platform, we used a general-purpose desktop computer with the following features:

- **CPU** : I7-2770
- **GPU**: non-existent
- **RAM**: 12Go clocked at 1333 MHz
- **Storage**: 256 GB
- **OS** : KDE Plasma 6.2
- **System type**: x64

and a LENOVO laptop for testing and writing:

- **CPU** : I5-4210U clocked at 1.7 GHz
- **GPU**: integrated graphic processeur
- **RAM**: 6 Go
- **OS** : Windows 8.1
- **System type**: x64

To test the fog nodes, we use the Raspberry PI 4 model B (Figure 3.1) and for the IoT object, we are using two NodeMCU (Figure 3.2) where specifications are given in Table 1.1.

The user's mobile application has been developed and tested on (OPPO F9 PRO), whose specifications are as follows [26]:

- **CPU**: Octa-core 2 GHz (4 Cortex-A73 & 4 Cortex-A53)
- **GPU**: Mali-G72 MP3
- **OS**: ColorOS 6.0.1 (Android 9)
- **RAM**: 4 GB
- **Screen**: 2280 x 1080 px (409 ppi density)
- **Storage**: 64GB

[1] blueprints that focus on structure and layout

Figure 3.1: Raspberry PI 4 model B

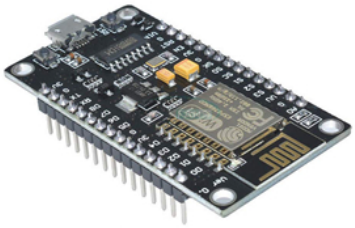


Figure 3.2: NodeMCU

3.4 Software tools

In this section, we present the software tools required to develop the access control protocol.

3.4.1 Arduino IDE

Arduino Integrated Development Environment (Arduino IDE) [27] is development software for programming on arduino boards. It facilitates development by providing code compilation and delivery to the board, code highlighting, plugins, libraries, auto-completion and installation of other boards. It is used to program the code for IoT objects (ESP-8266). To do this, we installed the ESP8266 board from the ESP8266 community in the board manager.

The language used for the development of ESP8266 is C++, a language widely used for programming microcontrollers, which allows good performance to be achieved. It is also standardized for firmware and kernel development.

The C++ used on microcontrollers is an adapted version. Some features of standard C++ are removed, such as threading and exceptions. Hardware-specific elements are added, such as serial ports and PINs-IO.

3.4.2 Android Studio

Android Studio [28] is software for developing applications on the Android platform. It is based on IntelliJ IDEA and provides tools such as code highlighting, autocompletion, refactoring tools, a user interface designer with a drag-and-drop interface, and the Gradle build system that handles compilation, packaging, and dependency management.

The language used to develop is Kotlin, which is a programming language developed by JetBrains and recommended for Android development. It makes it possible to build on top of Java code, with concise syntax, code security and improved asynchronous programming.

3.4.3 Node.js

"*Node.js*" is a runtime environment for executing JavaScript on the server side (outside the browser). It is built on the Chrome V8 engine, and is used for backend development, APIs, real-time applications, etc. Like Python, it provides a package manager, an integrated file system, an http bone, and asynchronous features.

For the administrator and the authority, we're using "Node.js" as the server to serve their web application, which is developed using "Solid.js" [29], which is a "Node.js" library for building user interfaces similar to "React.js".

The server API is built using a RESTful architecture. We use express.js which is a framework for building web and mobile applications in "Node.js". It is often described as the "de facto standard server framework for "Node.js".

3.4.4 GoLang

GoLang (go) is an open-source programming language developed by Google. It is used for web applications, cloud and network services...

3.4.5 Database

When choosing the database for the admin, authority, and fog node, we use (My Structured Query Language (MySQL), which is an open-source relational database management system.

- The admin stores data from fog nodes, IoT devices, and authorities in it.
- The fog node stores the data from the IoT devices in it.

The two databases must have the same system image.

For the mobile application, we are using *SQLite*, which is a lightweight relational database management system.

3.4.6 Used libraries

To develop our access protocol based on the MA-ABE model, we used the following libraries:

3.4.6.1 MA-ABE library

For MA-ABE, we use GO Functional Encryption (GoFE) which is a cryptographic library developed in GoLang and which offers various implementations of functional encryption schemes, including the MA-ABE scheme proposed by Allison Lewko and Brent Waters [30].

To enable the user to decrypt the MA-ABE encrypted token in the mobile phone, we use Go mobile, which is a sub-repository of goLang that allows us to compile code for mobile platforms (Android and iOS) and provides tools for creating mobile applications. We use it to compile the decryption code in the Android Archive file format, which can be used in Android studio.

To enable authorities to initialize the authority, create and renew attributes, and generate user keys, we create a main function in a separate '.go' file for each process, specify the arguments passed and the output returned, and then compile it into an executable as shown in Figure 3.3. We can now call the executable via any programming language.

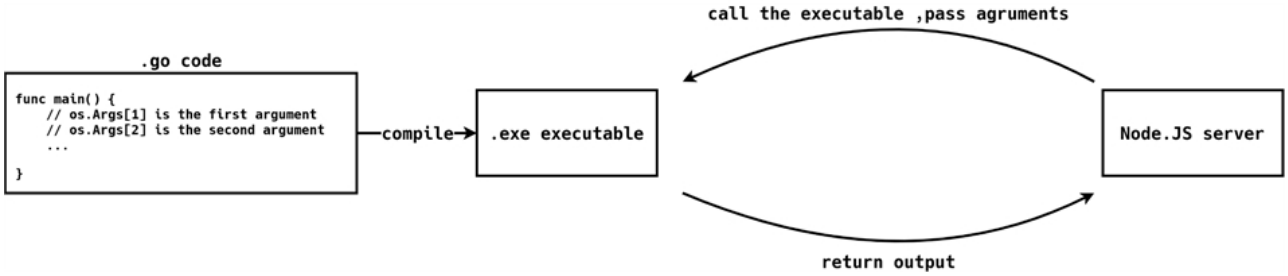


Figure 3.3: Process of manipulating the authorities

GoFE does not support a structured format for MA-ABE public parameters, authority public/private keys, cryptograms or user keys, it simply uses them as native GO objects. This works well within the same program, but does not work when the objects are transmitted over the network and manipulated by . To solve this problem, we have created JSON utility functions that translate objects into JSON structured format and vice versa (see Figure 3.4).

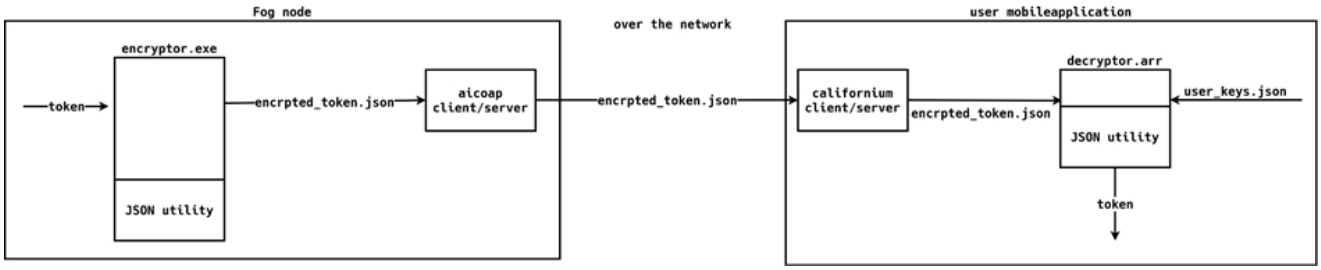


Figure 3.4: Creation of JSON utility functions

3.4.6.2 AIOCoAP protocol

Asynchronous IO Concurrent Application Protocol (AIOCoAP) [31] is a CoAP implementation in python. It implements the CoAP, OSCoRE and EDHOC protocols. It is used in the fog node for communication between it and IoT devices.

3.4.6.3 CoAP-simple-library

CoAP-simple-library [32] is a lightweight library that implements the basic functionality of CoAP in Arduino IDE/PlatformIO, ESP32 and ESP8266. It is used in the IoT object for communication with the fog node and the user's application.

3.4.6.4 Californium

Californium [33] is a library that implements the CoAP standard in Java. It is used in the user application for communication with fog nodes and IoT devices.

3.5 Software use in the system

In this section, we present the set of libraries for the platform. Figure 3.5 illustrates the interaction between the administrator (Admin) and the authorities, while Figure 3.6 shows the relationship between the administrator, the user application and the IoT objects managed in the system.

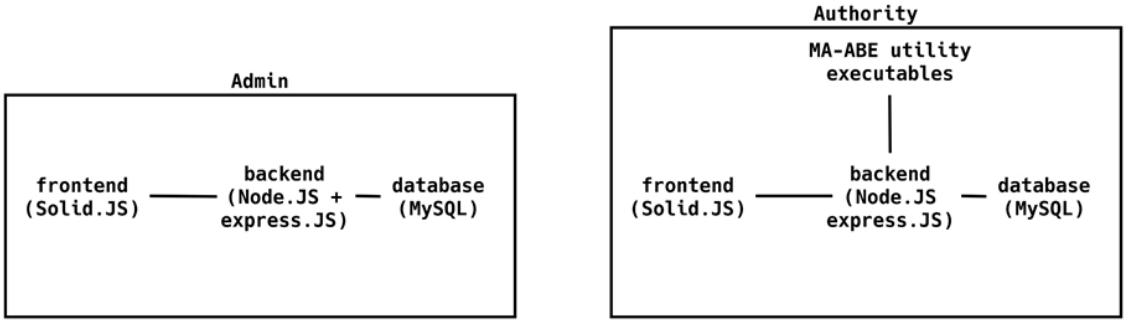


Figure 3.5: Interaction between the admin and the authorities

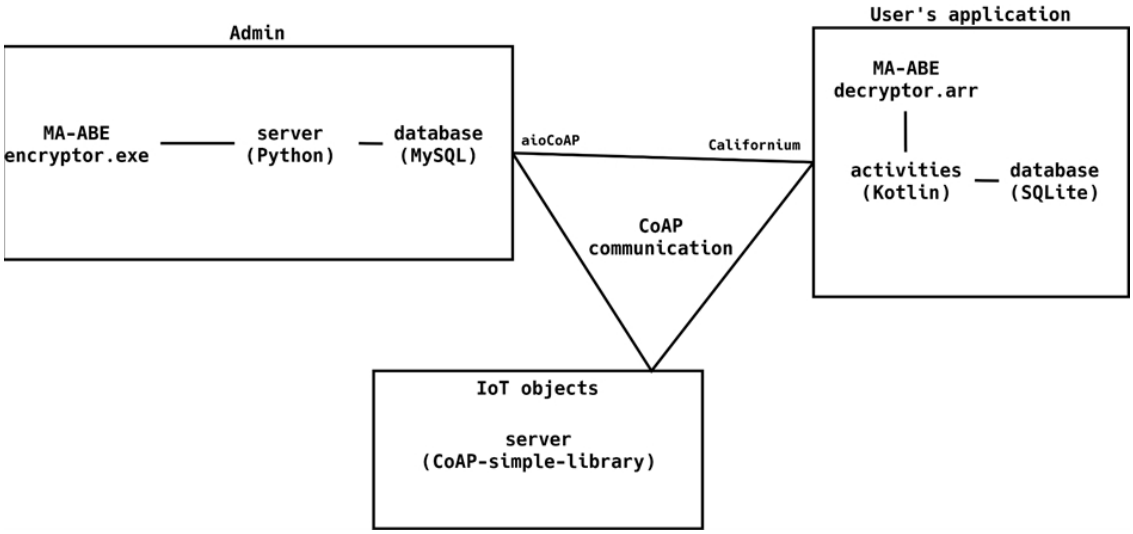


Figure 3.6: Relationship between the admin, the user application and the IoT objects

3.6 Presentation of the application

We have named the system ”SmarDustry”, which combines the words “Smart” because of the use of IoT to improve the intelligence of the system and ”Industry” because it is used in the

industrial environment.

3.6.1 Admin interface

The administration interface is divided into five screen pages:

1. **The home page**: Includes buttons for accessing the 4 branches as shown in Figure 3.7.

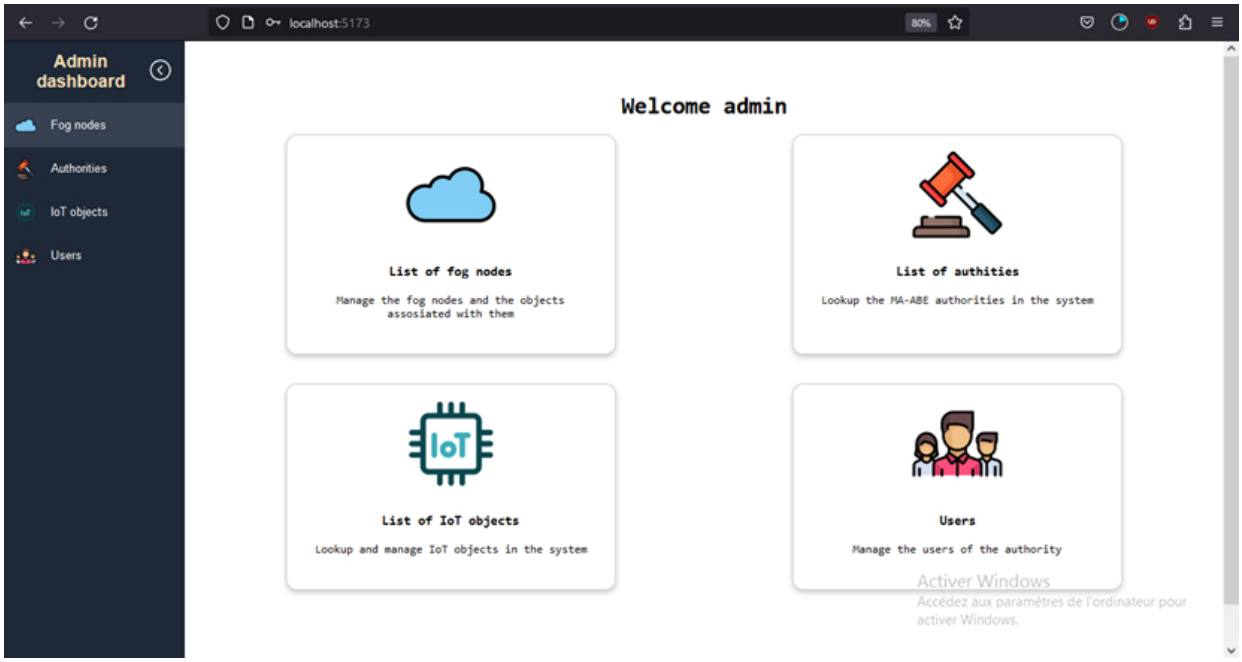


Figure 3.7: Home page of the admin

2. **The fog nodes page**: This page contains a list of registered fog nodes with their information, connectivity status and the configuration button used to configure their objects as depicted in Figure 3.8.

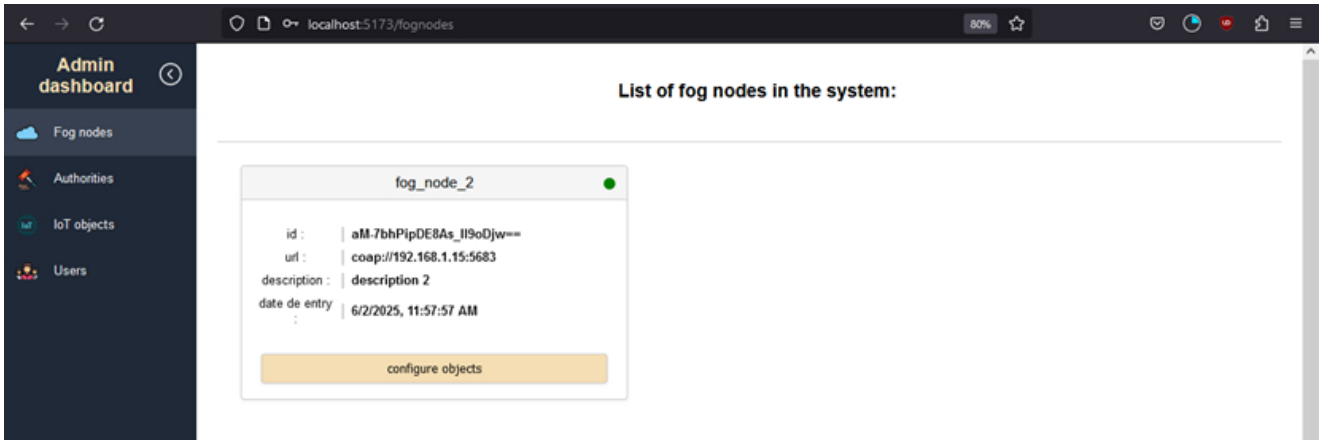


Figure 3.8: List of fog nodes page

3. **The fog node page**: This page contains information about the current fog node, a list of the objects stored in it and a form for adding new objects to the fog node as illustrated in Figure 3.9.

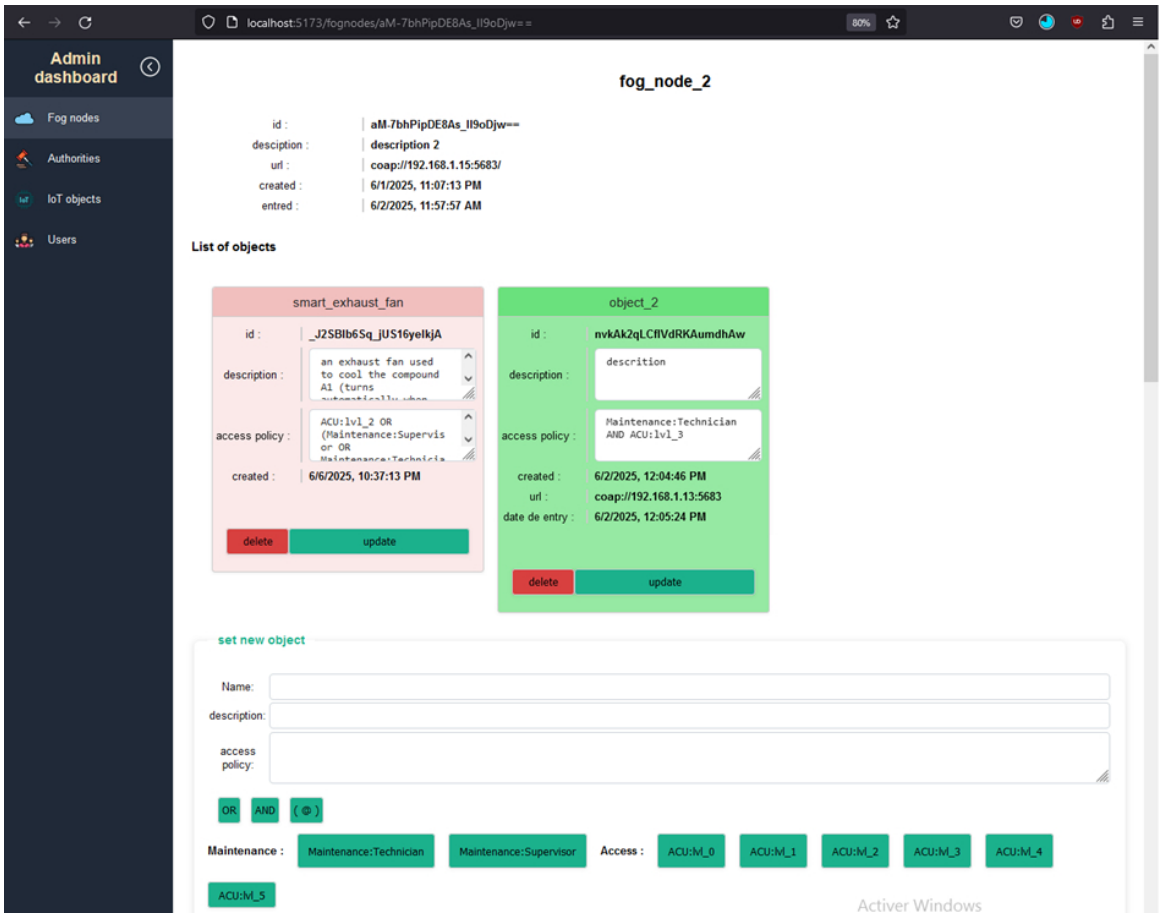


Figure 3.9: Fog node page

4. **Authorities page**: It contains a list of the authorities registered in the system, together with information on their attributes as shown in Figure 3.10.

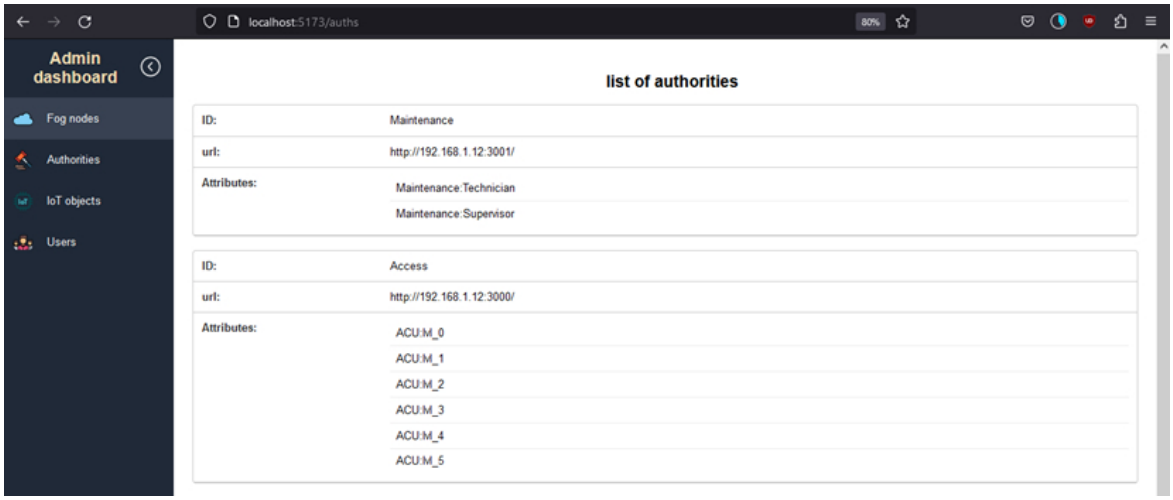


Figure 3.10: Authorities page

5. **Users page**: This page contains a list of system users as shown in Figure 3.11. It also contains a form for adding new users.

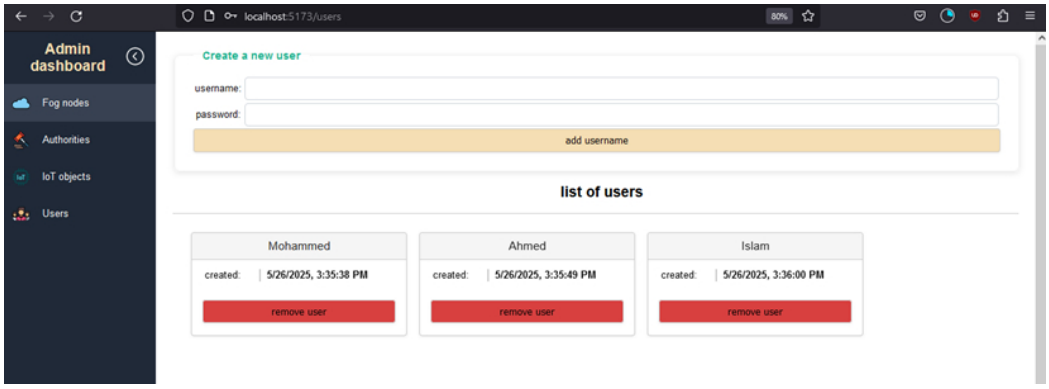


Figure 3.11: Users page

3.6.2 Authority interface

The home page of the authority interface contains two branches as presented in Figure 3.12:

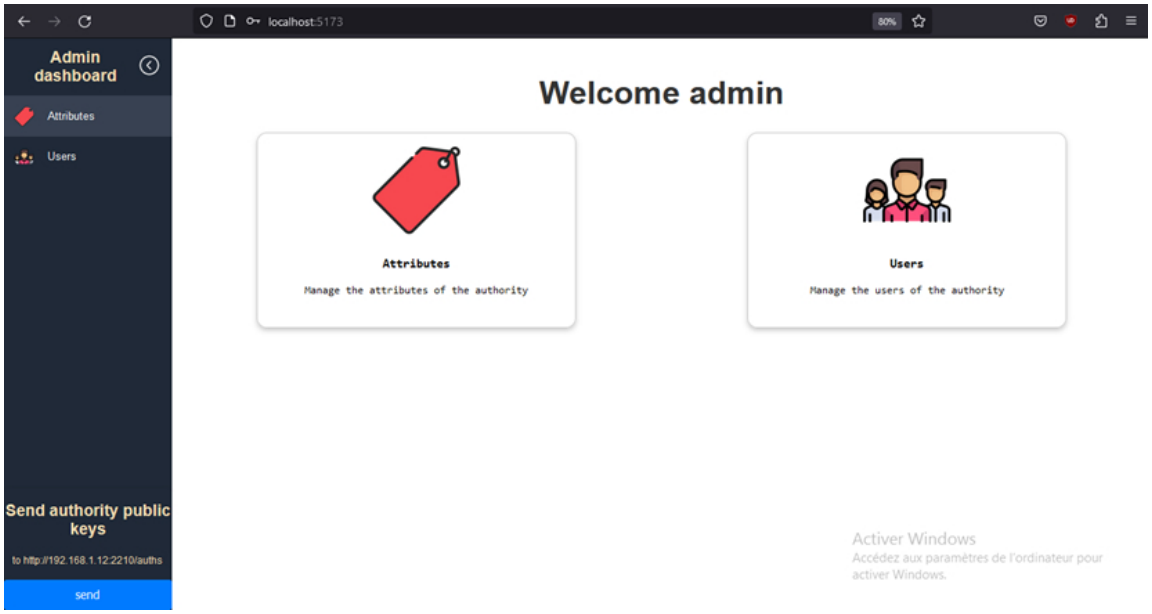


Figure 3.12: Authority interface

1. **The attributes page**: This page contains a list of the authority's attributes. For each attribute, we can see the name of the attribute, the "Renew key" button, the "Remove attribute" button and the public key. The page also contains a form for creating a new attribute for the authority. Figure 3.13 illustrates the different operations that can be performed on authority attributes.

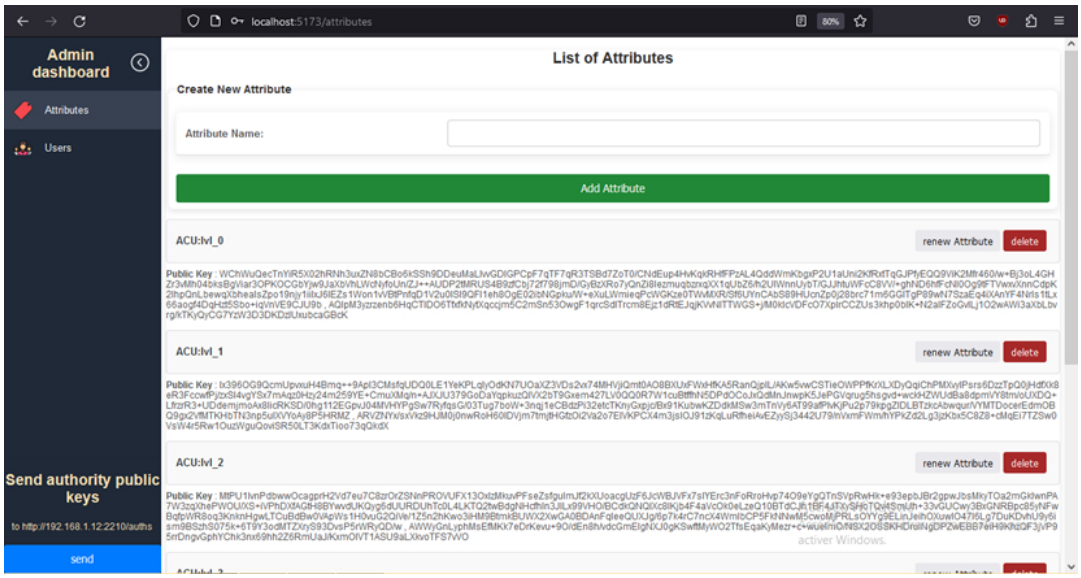


Figure 3.13: Attribute handling page

If the public key does not exist on the server (when the server is initialised by the authority administrator), the page asks for the JSON file of the public parameters as shown in Figure 3.14.

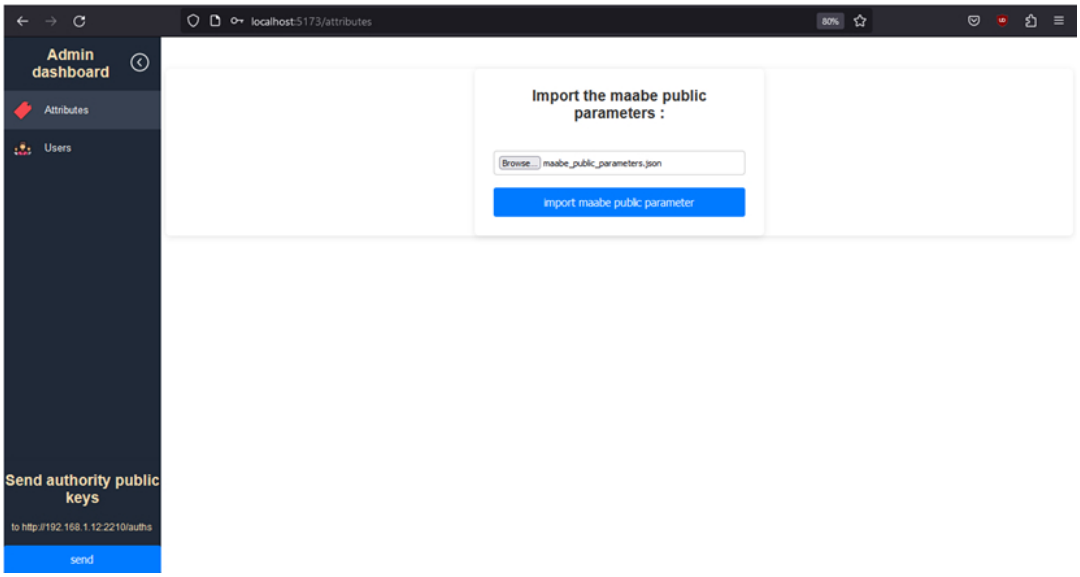


Figure 3.14: Import of public parameters

Moreover, if there is no authority file on the server. The page displays a form for importing an authority JSON file or creating a new authority as depicted in Figure 3.15

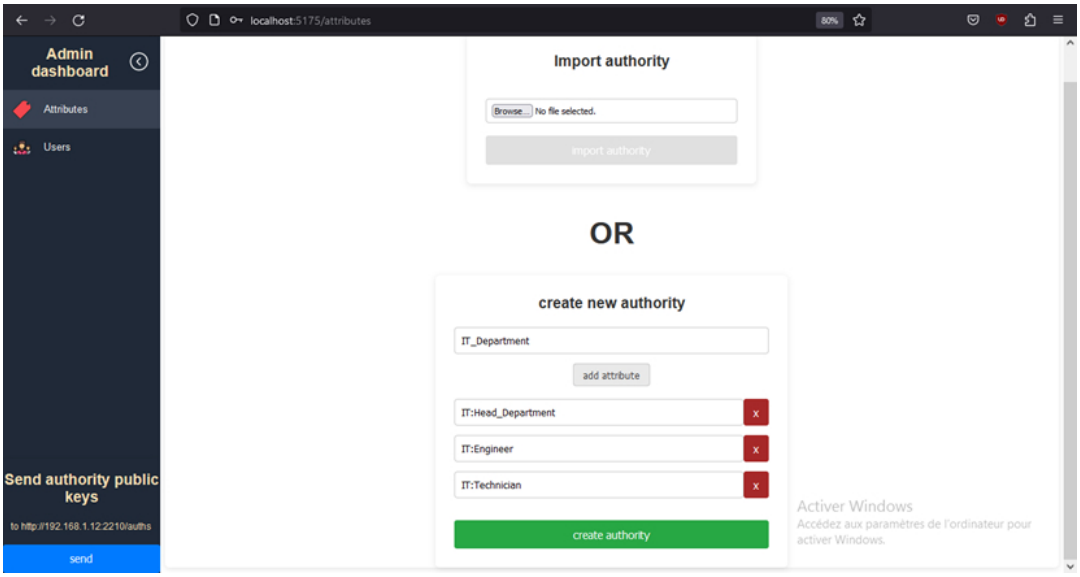


Figure 3.15: Creation of new authority page

2. **The users page**: This page contains the list of users. For each user, it displays the list of attributes, the "Add an attribute" button, the "Remove an attribute" button and the "Remove" button. The page also includes a button for importing users from the administration server as presented in Figure 3.16.

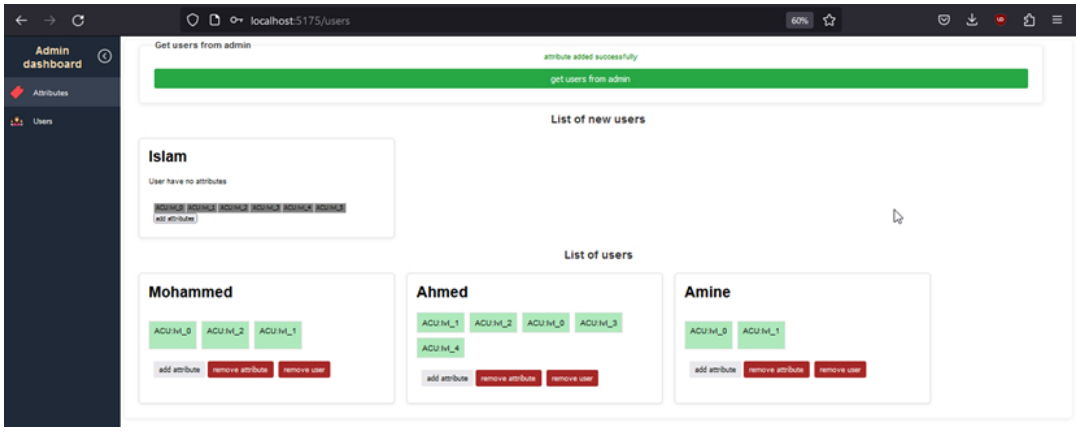


Figure 3.16: Users handling page

3.6.3 User's application

In the user application part, we have several interfaces represented by different pages:

1. **Login page**: This page contains a form for user credentials and the fog node's IP address, as well as a "Continue" button as shown in Figure 3.17.

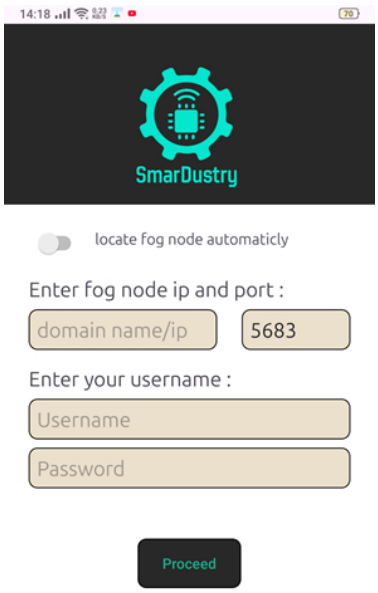


Figure 3.17: Login page for the user's application

2. **Home page**: This page contains the fog node status, authority buttons and the objects page as depicted in Figure 3.18.

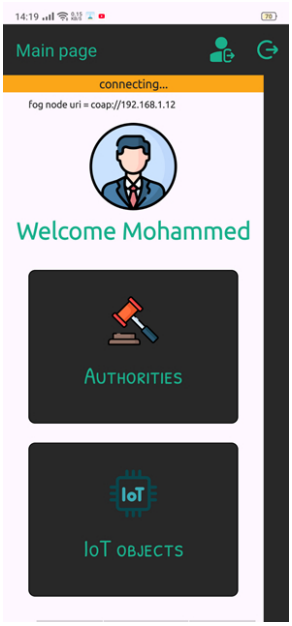


Figure 3.18: Home page

3. **Authorities page**: This page contains a list of authorities. The user can retrieve the keys by clicking on *"get keys"*. If the user already has the keys, the authority displays the attributes that it has as shown in Figure 3.19.

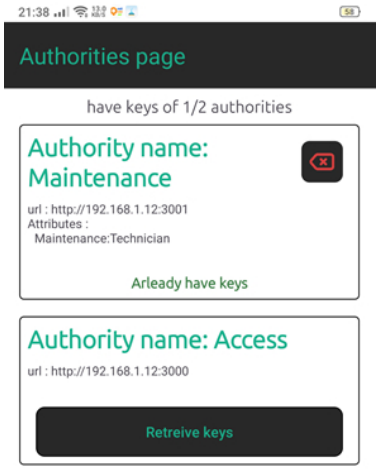


Figure 3.19: Authorities page

4. **IoT object list page**: This interface contains a list of objects controlled by the fog node. For each object, if the object has not entered the network, it is marked as offline; if it has, it displays the *"get access* button, which requests the encrypted token and attempts to decrypt it. If successful, the status of the object turns green and *"get access"* is replaced by *"access object"* as illustrated by the Figure 3.20.

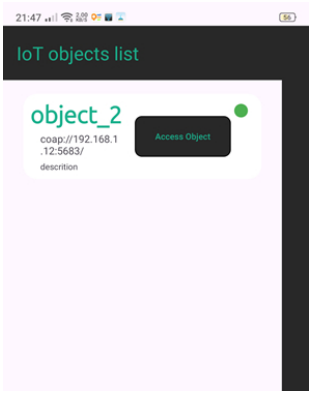


Figure 3.20: IoT object list page

5. **IoT object page**: This page contains the object's control panel. Each object has its own customised interface for monitoring and controlling the defined object.

3.7 Scenario-based testing

3.7.1 Initializing the fog node

First, the administrator SSH connects to a fog node terminal, downloads the fog node code from a repository, accesses its folder, then sets variables and downloads dependencies using a shell install script, as shown in the Figure 3.21.

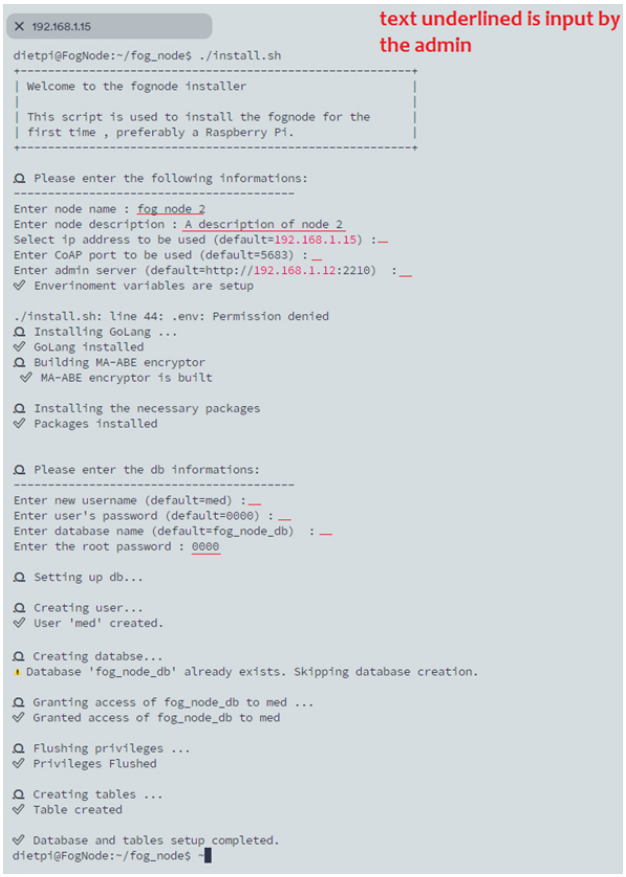


Figure 3.21: Fog node initialization via *"./install.sh"*

The admin then starts the fog node using the fog node terminal (ssh), as shown in Figure 3.22.

Figure 3.22: Starting the Fog node using python3 *"server.py"*

3.7.2 Adding IoT objects to the Fog node

The admin can now add IoT objects to this fog node via their admin interface and when the IoT object is successfully added, the add IoT object interface is displayed as shown in Figure 3.23.

Figure 3.23: Page for adding an IoT object

Users still cannot access the object because it is not yet connected to the network. When the object starts up, it retrieves the token from the fog node, as shown in Figure 3.24.

Figure 3.24: Token request by IoT object at Fog node

The fog node that receives the object request generates a token for it and sends it to the requesting IoT object as shown in Figure 3.25.

Figure 3.25: Generation of a token by the Fog node

3.7.3 Connecting a user to IoT objects

The user can view the object in their application and try to access it, as shown in Figure 3.26.

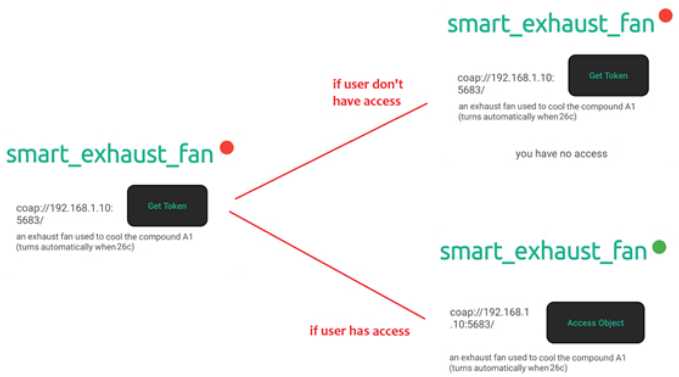


Figure 3.26: View the object via the application

If it is available, the user accesses the object's page, where they can monitor the object and perform operations on it. An example of a page is shown in the Figure 3.27.

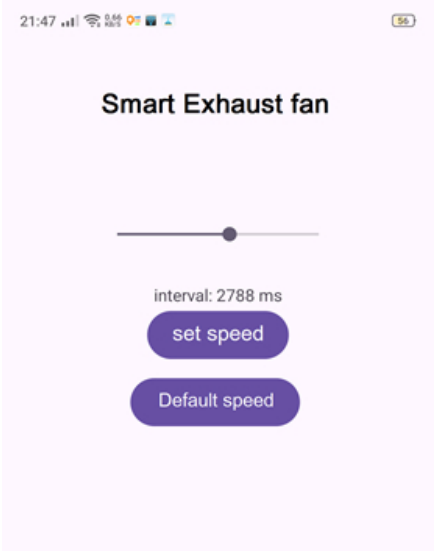


Figure 3.27: Access to the IoT object

3.8 Conclusion

To illustrate how our application works, in this chapter we have presented the different tasks that our application can perform in a secure way.

The platform developed enables Fog nodes to be subscribed to by the admin, IoT objects to be connected to Fog nodes and IoT objects to be accessed by users. All tasks are performed using robust and effective security mechanisms.

GENERAL CONCLUSION

This project highlights the possibility of using cryptography, in particular MA-ABE, as a means of a decentralised, stateless access control mechanism, where authorities retain their independence from each other and are not constrained by a central node. It also shows an example of its use in an industrial environment.

The system presents some difficulties, in particular the problem of user authentication in a decentralised system where users only have to authenticate once.

The system is functional. Nevertheless, it can be improved to become a system that can be integrated into companies and factories. Additions such as logging integration, improved user interface, password recovery, improved certificate distribution, robust error handling and quality of life improvements.

There will also be a number of additions to improve the system, such as the option of giving users temporary access.

References

BIBLIOGRAPHY

[1] GeeksforGeeks. *Architecture of Internet of Things (IoT)*. https://www.geeksforgeeks.org/computer-networks/architecture-of-internet-of-things-iot/. Last Accessed: 2025-06-13. 2025.

[2] Kinza Yaser and Sharon Shea. *IoT security*. https://www.talend.com/fr/resources/iot-definition/. Accessed: 2025-05-08. 2025.

[3] Nokia OYJ. *Nokia Threat Intelligence Report 2023*. Nokia.

[4] Ahmed Sultan. *Summarize Fundamental Security Concepts*. Tech. rep. Netriders Academy, 2023.

[5] Sabrina De Capitani di Vimercati. "Discretionary Access Control Policies (DAC)." In: *Encyclopedia of Cryptography and Security*. Ed. by Henk C. A. van Tilborg and Sushil Jajodia. Boston, MA: Springer US, 2011, pp. 356–358. DOI: 10.1007/978-1-4419-5906-5_817. URL: https://doi.org/10.1007/978-1-4419-5906-5_817.

[6] Bhavani Thuraisingham. "Mandatory Access Control." In: *Encyclopedia of Database Systems*. Ed. by LING LIU and M. TAMER ÖZSU. Boston, MA: Springer US, 2009, pp. 1684–1685. ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_214. URL: https://doi.org/10.1007/978-0-387-39940-9_214.

[7] Ravi S. Sandhu. "Role-based Access Control." In: ed. by Marvin V. Zelkowitz. Vol. 46. Advances in Computers. Elsevier, 1998, pp. 237–286. DOI: https://doi.org/10.1016/S0065-2458(08)60206-5. URL: https://www.sciencedirect.com/science/article/pii/S0065245808602065.

[8] Vincent C. Hu et al. "Attribute-Based Access Control." In: *Computer* 48.2 (2015), pp. 85–88. DOI: 10.1109/MC.2015.33.

[9] Melissa Chase. "Multi-authority Attribute Based Encryption." In: *Proceedings of the 4th International Conference on Theory of Cryptography (TCC 2007)*. Ed. by Salil P. Vadhan. Springer. Amsterdam, The Netherlands: Springer Berlin Heidelberg, 2007, pp. 515–534.

[10] Cullen Jennings, Jonathan Schoenwaelder, and Thomas Herbert. *Terminology for Constrained-Node Networks (RFC 7228)*. Tech. rep. Internet Engineering Task Force (IETF), 2014.

[11] espressif, ed. *ESP32 Series Datasheet (v4.9)*. 2025.

[12] espressif, ed. *ESP8266EX Datasheet (v7.0)*. 2023.

[13] Netriders Academy. *Raspberry Pi 4 Tech Specs*. `https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/`. Last access 12 April 2025.

[14] Arduino ESP8266 Read-the-docs. *BearSSL WiFi Classes*. `https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/bearssl-client-secure-class.html`. Last access 10 April 2025.

[15] Sachin Babar et al. "Proposed embedded security framework for Internet of Things (IoT)." In: *Proceedings of the 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*. IEEE. Chennai, India, 2011, pp. 1–5. DOI: 10.1109/WIRELESSVITAE.2011.5940923.

[16] Michelle Rossevelt. *How Much Does It Cost To Encrypt Data?* `https://www.newsoftwares.net/blog/how-much-does-it-cost-to-encrypt-data/`. Last access 12 April 2025.

[17] Concurrency. *Why HTTP is not suitable for IOT applications*. `https://concurrency.com/blog/why-http-is-not-suitable-for-iot-applications/`. 2019.

[18] Aaron. *Is UDP the right choice for IoT*. `https://medium.com/@webtolife/is-udp-the-choice-for-iot-e230d069eae5/`.

[19] HiveMQ Team. *Introducing the MQTT Protocol – MQTT Essentials: Part 1*. `https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt/`.

[20] Jonathan Beri. *A Field Guide to CoAP — Part 1*. `https://jonathanberi.medium.com/a-field-guide-to-coap-part-1-75576d3c768b/`. 2020.

[21] Francesca Palombini, Göran Selander and John Preu Mattsson. *OSCORE: A look at the new IoT security protocol*. `https://jonathanberi.medium.com/a-field-guide-to-coap-part-1-75576d3c768b/`. Last access 7 November 2019.

[22] Dan Boneh and Matt Franklin. “Identity-Based Encryption from the Weil Pairing.” In: *Proceedings of 21st Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2001*. Ed. by Joe Kilian. Springer. Santa Barbara, California, USA: Springer Berlin Heidelberg, 2001, pp. 213–229.

[23] Susan Hohenberger and Brent Waters. “Online/Offline Attribute-Based Encryption.” In: *Proceedings of the 17th International Conference on Practice and Theory in Public-Key Cryptography*. Ed. by Hugo Krawczyk. Springer. Buenos Aires, Argentina: Springer Berlin Heidelberg, 2014, pp. 293–310.

[24] K.M Ashwini and B.S Umashankar. “A Review on Attribute Based Encryption (ABE) and ABE Types.” In: *International Journal of Computer Science and Mobile Computing* 5.5 (2016), pp. 142–146.

[25] Brien Posey and Sharon Shea. *What is fog computing?* `https://www.techtarget.com/iotagenda/definition/fog-computing-fogging`. Last access 22 October 2021. 2021.

[26] GSM Arena. *Oppo F9 (F9 Pro)*. `https://www.gsmarena.com/oppo_f9_(f9_pro)-9286.php`. Last access 12 May 2025.

[27] arduino. *Arduino Editor*. `https://www.arduino.cc/en/software/`. Last access 12 May 2025.

[28] Android Studio. *Android Studio*. `https://developer.android.com/studio/intro?hl=fr`. Last access 10 June 2025.

[29] Solid.js. *Solid.js*. `https://www.solidjs.com/`. Last access 25 May 2025.

[30] Allison Lewko and Brent Waters. “Decentralizing Attribute-Based Encryption.” In: *Proceedings of the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Ed. by Kenneth G. Paterson. Springer. Tallinn, Estonia: Springer Berlin Heidelberg, 2011, pp. 568–588.

[31] Christian Amsüss. *chrysn/aiocoap*. `https://github.com/chrysn/aiocoap`. Last access 25 February 2025.

[32] Hirotaka Niisato. *hirotakaster/CoAP-simple-library*. `https://github.com/hirotakaster/CoAP-simple-library`. Last access 20 March 2021.

[33] Californium. *eclipse-californium/californium*. `https://github.com/eclipse-californium/californium`. Last access 20 February 2025.

Business Model Canvas (BMC)

جامعة أبو بكر بلقايد
UNIVERSITY OF TLEMCEN

Business Model Canevas : BMC

Partenaires clés
Key Partnerships
الشراكة الرئيسية

- Incubateurs
- Importateurs des équipements électroniques et objets IoT

Activités clés
Key Activities
الأنشطة الرئيسية

- Installation de système de gestion de contrôle décentralisé

Ressources clés
Key resources
الموارد الرئيسية

- 2 PCs, 2 PCs portable, un bureau, loyer, un véhicule commercial
- Équipe de développement et équiper d'installation

Proposition de valeur
Value Proposition
القيمة المقترحة

- Une solution de problème de gestion du contrôle d'accès aux machines, équipements et aux zones dans l'environnement industriel.
- Elle permet aux autorités dans une entreprise de contrôler l'accès des employés aux équipements.
- Elle permet aussi aux employés autorisés de surveiller et d'opérer les machines.

Relation clients
Consumer Relationship
علاقة مع العملاء

- Une relation B2B avec les entreprises
- B2B : une relation avec les sociétés étatiques

Canaux de distribution
Channels
قنوات التوزيع

- Site web professionnel
- Emails envoyé aux entreprises potentiels
- Une présence dans les conférences sur l'innovation industrielle.

Segment client
Customer Segment
انواع العملاء

- Entreprises qui travailent avec équipements coûteux et sensibles
- Entreprises nécessitent une solution de contrôle à distance.
- Entreprises nécessitent une solution de contrôle d'accès
- Les usines
- Les sociétés générales

Coûts : Cost sructure : التكاليف

Les salaires des employés (programmeurs, personnel du service client) constituent un coût fixe, tandis que les coûts variables correspondent aux coûts des matériaux importés.

- **Première année** :
  - 20 Raspberry Pi 4 Model B (4GB RAM) = 156 200 DA
  - Matériel réseau et installation = 153 500 DA
  - 2 × PC portables = 177 500 DA
  - Serveur DELL PowerEdge R230 = 235 000 DA
  - Équipe de développement (3 personnes) = 55 000 DA/mois × 12 = 660 000 DA
  - Équipe d'installation (2 personnes) = 45 000 DA/mois × 12 = 540 000 DA
  - Location de voiture commerciale = 70 000 DA/mois × 12 = 840 000 DA
  - Location de bureau = 50 000 DA/mois × 12 = 600 000 DA
  - Site web = 2 000 DA/mois × 12 = 24 000 DA

  Total première année = 2 466 700 DA

- **Deuxième année :**
  50 Raspberry Pi 4 Model B (4GB RAM) = 390 500 DA
  Matériel réseau = 41 200 DA
  Équipe de développement (4 personnes) = 55 000 DA/mois × 12 = 660 000 DA
  Équipe d'installation (5 personnes) = 45 000 DA/mois × 12 = 2 700 000 DA
  Achat voiture commerciale (nouvelle) = 2 555 000 DA
  Location de bureau = 50 000 DA/mois × 12 = 600 000 DA
  Site web = 2 000 DA/mois × 12 = 24 000 DA

  Total deuxième année = 6 970 700 DA

Total troisième année = 4 811 900 DA

Revenus (Revenue) :مصادر الدخل

- La vente l'installation du système de contrôle d'accès
- Services de conseil technique
- Maintenance de système
- **Première année** :
  - Système complet (matériel + logiciel) : 115 000 – 175 000 DA
  - Installation sur site : 25 000 – 35 000 DA
  - Abonnement annuel de maintenance : 50 000 DA
  - Revenu total par client = 190 000 – 260 000 DA
  - Estimation pour 10 clients = 1 900 000 – 2 600 000 DA
  - Résultat net (perte) = –1 666 700 DA
- **Deuxième année** : Résultat net = +2 979 300 DA
- **Troisième année** : Résultat net = +9 288 100 DA