People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
Echahid Cheikh Larbi Tebessi University – Tebessa
Faculty of Exact Sciences and Natural and Life Sciences
Department of Computer Science

Thesis

Submitted in partial fulfillment of the requirements for the
Master's Diploma (LMD).
In: Networks and IT security

Theme

Secure communication system in an IoT environment

Present by:
ATTIA Meriem

Before the jury composed of:

Dr. HAOUAM Mohamed Yassine	**MCA**	Echahid Cheikh Larbi Tebessi University	**President**
Dr. MEKHAZNIA Taher	**MCA**	Echahid Cheikh Larbi Tebessi University	**Examiner**
Dr. SAHRAOUI Abdelatif	**MCA**	Echahid Cheikh Larbi Tebessi University	**Framer**

Publicly sustained on:

09/06/2024

Secure communication system in an IoT environment

ATTIA Meriem

Echahid Cheikh Larbi Tebessi University

Thanks:

I would like to express my gratitude to Allah, the Almighty and Merciful, for granting me the strength and patience to complete this work successfully. A huge thank you to my parents for their unwavering support and constant efforts. Their presence and encouragement have been a source of inspiration throughout this project, and I am deeply grateful to them.

I would also like to warmly thank **Dr. Sahraoui Abdelatif** for his valuable advice and wise educational guidance. Likewise, I am grateful to all the teachers who passed on their knowledge to me and gave me valuable support in my studies.

I would like to express my sincere gratitude to the members of the jury for the attention they paid to my work, as well as for their valuable contributions which enriched this research.

Finally, a big thank you to all the people who contributed, directly or indirectly, to the completion of this work. Your support and collaboration have been invaluable, and I extend to you all my gratitude and deep respect.

Thank you all very much.

Dedication:

To my mother and my father.

To my sister and my brother.

To all the precious members of my extended family.

To my supervisor Dr. SAHRAOUI Abdellatif.

To my friends.

ATTIA Meriem

Abstract:

With the rapid rise of Internet of Things (IoT) devices, new opportunities are emerging, but this also brings significant data security and privacy challenges. Securing data from these devices has become a priority, requiring innovative and robust solutions. To guarantee the protection and confidentiality of the information exchanged, one approach proposes the use of Blockchain technology and smart contracts to set up a secure communication system in the IoT.

The purpose of this study is the establish a secure communication system for IoT environments using Blockchain technology and smart contracts. The main objective is to ensure the confidentiality, integrity, and authenticity of data exchanged between IoT devices while respecting the limited resource constraints of these devices. To do this, several smart contracts have been developed to manage data access, storage, and verification. Experimental results demonstrate that the use of Blockchain and smart contracts significantly improves data security by providing trust and transparency mechanisms, thereby overcoming current security challenges in IoT environments.

Keywords:

Internet of Things (IoT), communication system, security, Blockchain, smart contract.

Résumé :

Avec l'essor rapide des appareils de l'Internet des objets (IoT), de nouvelles opportunités émergent, mais cela entraîne également des défis significatifs en matière de sécurité des données et de protection de la vie privée. La sécurisation des données provenant de ces dispositifs est devenue une priorité, nécessitant des solutions innovantes et robustes. Pour garantir la protection et la confidentialité des informations échangées, une approche propose l'utilisation de la technologie Blockchain et des contrats intelligents pour mettre en place un système de communication sécurisé dans l'IoT.

L'objet de cette étude propose une mise en place d'un système de communication sécurisé pour les environnements de l'IoT en utilisant la technologie de la Blockchain et des contrats intelligents. L'objectif principal est d'assurer la confidentialité, l'intégrité et l'authenticité des données échangées entre les dispositifs IoT, tout en respectant les contraintes de ressources limitées de ces dispositifs. Pour ce faire, plusieurs contrats intelligents ont été développés pour gérer l'accès, le stockage et la vérification des données. Les résultats expérimentaux démontrent que l'utilisation de la Blockchain et des contrats intelligents améliore considérablement la sécurité des données en offrant des mécanismes de confiance et de transparence, ce qui permet de surmonter les défis actuels en matière de sécurité dans les environnements IoT.

Les mots clé:

The Internet of Things (IoT), communication systems, security, Blockchain, and smart contracts.

الملخص

مع الارتفاع السريع لأجهزة إنترنت الأشياء (IoT)، تظهر فرص جديدة، ولكن هذا يجلب أيضًا تحديات كبيرة فيما يتعلق بأمن البيانات والخصوصية. لقد أصبح تأمين البيانات من هذه الأجهزة أولوية، الأمر الذي يتطلب حلولاً مبتكرة وقوية. ولضمان حماية وسرية المعلومات المتبادلة، يقترح أحد الأساليب استخدام تقنية Blockchain والعقود الذكية لإنشاء نظام اتصالات آمن في إنترنت الأشياء.

الغرض من هذه الدراسة هو اقتراح إنشاء نظام اتصالات آمن لبيئات إنترنت الأشياء باستخدام تقنية Blockchain والعقود الذكية. الهدف الرئيسي هو ضمان سرية وسلامة وصحة البيانات المتبادلة بين أجهزة إنترنت الأشياء، مع احترام قيود الموارد المحدودة لهذه الأجهزة. وللقيام بذلك، تم تطوير العديد من العقود الذكية لإدارة الوصول إلى البيانات وتخزينها والتحقق منها. تظهر النتائج التجريبية أن استخدام بلوكتشين والعقود الذكية يحسن بشكل كبير أمن البيانات من خلال توفير آليات الثقة والشفافية، وبالتالي التغلب على التحديات الأمنية الحالية في بيئات إنترنت الأشياء.

الكلمات المفتاحية:
انترنت الأشياء، نظام اتصال، الأمان، البلوكشين، العقود الذكية.

Table of contents

List of Figures

List of tables

General introduction

Introduction:

With the rapid rise of the Internet of Things (IoT), our world is becoming increasingly interconnected, facilitating unprecedented communication and interaction between devices. From smart homes to connected cities, including industrial applications, IoT is transforming our daily lives by offering a multitude of new opportunities. However, this proliferation of interconnected devices also poses significant security challenges. IoT systems are particularly vulnerable to cyberattacks due to their often decentralized architecture, technological diversity, and limited processing capacity.

In this context, securing communications within IoT environments becomes crucial. Data exchanged between IoT devices is often sensitive and critical, ranging from personal information to control data for industrial systems. Therefore, it is imperative to ensure the confidentiality, integrity, and authenticity of these communications to prevent intrusions, interceptions, and data manipulation.

This work proposes the development of a secure communication system adapted to IoT environments using Blockchain technology and smart contracts. We will explore the specific challenges posed by security in the context of IoT and propose innovative solutions to overcome them. Our objective is to design and implement a communication protocol that ensures a high level of security while being adapted to the constraints of IoT devices, particularly in terms of computing power, energy consumption, and bandwidth.

Problematic:

Our problem is to find solutions to guarantee data security in IoT environments, taking into account the particularities of IoT technology as well as the resource limits of IoT devices, while allowing efficient use of Blockchain technology. (BC) and smart contracts. More specifically, we ask ourselves the following questions:

- What cryptography techniques can be used to secure IoT communications without imposing excessive computational and energy overhead?

- What specific solutions can be developed to overcome the current limitations of IoT devices in terms of security and data management?

- How to ensure effective cryptographic key management in a decentralized IoT environment?

Objective:

The objective of our work is to design and implement a secure communication system for IoT environments, ensuring the confidentiality, integrity, and authenticity of the data exchanged while respecting the resource constraints of IoT devices. To achieve this, we will analyze the vulnerabilities and threats specific to IoT devices and develop a secure communication protocol. Effective cryptographic key management will also be offered.

Memory plan:

Our dissertation is structured into four chapters which address different aspects of our research on the secure communication system based on Blockchain and smart contracts in an IoT environment.

- **Chapter 1:** Introduction to IoT environments.
- **Chapter 2:** State of the art of security mechanisms for communication in an IoT environment.
- **Chapter 3:** Architecture based on Blockchain and smart contracts for communication in an IoT environment.
- **Chapter 4:** Implementation and evaluation.
- **General conclusion.**
- **References.**

Chapter 1:

Introduction to IoT environments.

1. Introduction :

The Internet of Things (IoT) is a revolutionary concept referring to the interconnection of physical objects via the Internet, enabling communication and data exchange between these objects and other connected systems. IoT encompasses a wide range of applications and industries, from smart homes to smart cities, healthcare, agriculture, logistics, and more.

In this first chapter, we will present the fundamental concepts of an IoT environment, addressing its definition, its architectures, its communication methods and protocols, its applications, as well as the problems and attacks that these environments can face.

2. IoT environments:

2.1.Definition :

The Internet of Things (IoT) is a network of physical objects: devices, equipment, vehicles, buildings, and other elements embedded with electronics, circuits, software, sensors, and network connections that enable these objects to collect and exchange data.

The Internet of Things allows objects to be sensed and controlled remotely through existing network infrastructure, creating opportunities for more direct integration of the physical world into computing systems, thereby increasing efficiency and accuracy.[1]

2.2. History:

The concept of IoT dates back to 1982. At the time, machines modified to make cola were connected to the Internet, allowing them to report the drinks they contained and determine whether the drinks were cold.

Then, in 1991, Mark Weiser first presented the modern vision of IoT in the form of ubiquitous computing. However, Bill Joy mentioned communication between devices in his 1999 Internet Taxonomy. The same year, Kevin Ashton proposed the term

“Internet of Things” describes a system of interconnected devices.[2]

3. IoT architecture:

The IoT architecture is made up of several interconnected systems organized in layers, each performing a specific function to ensure the proper functioning of the whole. There is no single standard reference architecture for IoT because it encompasses a variety of technologies.

In most cases it is made up of 4 constituent blocks:

Scalability, Functionality, Availability, and Maintainability.

The most basic and widely accepted format is a 4-layer IoT architecture [3]:

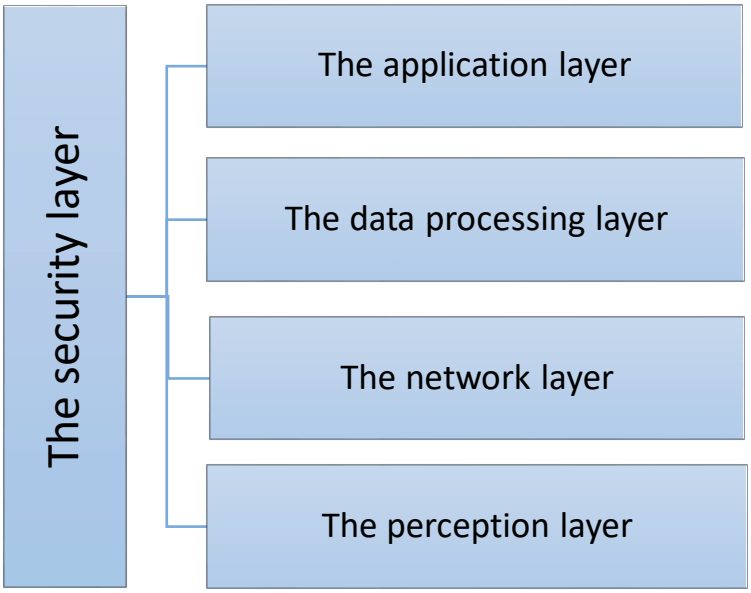

Figure 1.1: IoT architecture.

3.1. The perception layer:

This layer is responsible for converting analog signals into digital data and vice versa. IoT objects can be classified into the following groups:

Sensors: probes, counters, counters, etc. It records physical parameters such as temperature and humidity, converts them into electrical signals, and sends them to IoT systems. IoT sensors are generally small and consume very little power.

Actuator: This converts electrical signals from IoT systems into physical movement. Actuators are used in motor controllers, lasers, and robotic arms.

Machines and equipment: connected to sensors and actuators or connected as an integrated component.

3.2. The network layer:

This layer is responsible for transmitting the data received from the perception layer to the database using connectivity technologies.

Communication protocols include short-range protocols such as Wi-Fi and BLE, and long-range protocols such as LoRa, 3G/4G, and NBIoT.

3.3. The data processing layer:

The processing layer collects, stores, and processes data from previous layers. All of these tasks are usually managed through an IoT platform and include two main steps.

Data accumulation stage: Real-time data is collected and made available via the API to meet the

needs of real-time applications. This step defines, among other things, whether the data is relevant to your business needs and where the data should be located.

Data abstraction stage: The data is now ready and can be used by consumer applications to generate insights. The combined data accumulation and abstraction steps hide hardware details and improve the interoperability of smart devices.

3.4. The application layer:

The application layer is responsible for interacting with end users and providing application-specific services.

It can be built directly on the IoT platform, which provides advanced analytics and data visualization, or use APIs to integrate previous layers.

Security layers cover all previous layers and are essential to ensure IoT security. Although there is no consensus on the number of layers in an IoT architecture, a four-layer architecture is the most commonly used and accepted.

4. IoT communication modes:

4.1. Wireless mode:

- **WiFi :**

Wi-Fi technology uses radio waves to allow devices to communicate. Wi-Fi has a wider range than Bluetooth, making it more vulnerable to attack and hacking by attackers.

Wi-Fi is not the Internet itself, it simply provides wireless connectivity between devices. Outdoors, Wi-Fi can transmit data up to 100 meters, but only up to 20 meters in the presence of obstacles. [4]

- **Bluetooth:**

Bluetooth technology is also used in IoT. Use this technology when you need interaction between devices within 10 meters.

Bluetooth technology uses radio waves to be able to communicate in any direction. Bluetooth devices have low power consumption, allow devices to connect nearby, and can connect up to eight devices at the same time. [5]

- **ZigBee** :

Zigbee is a wireless technology enabling communication between devices, and operates according to the IEEE802.15.4 standard, resulting in low power consumption and low cost. Zigbee has low power consumption, and long battery life, and plays an important role in IoT. IEEE802.15.4 facilitates communication over various network topologies, such as mesh and star topologies, and these Networks can be extended at any time by adding other edge or node networks.[6]

- **NFC:**

Near field communication is a set of short-range wireless technologies at 13.56 MHz, typically requiring a distance of 4 cm. Allows intuitive initialization of wireless networks, NFC complete Bluetooth, and 802.11 with long-range capabilities up to a distance of approximately 10 cm. It works in dirty environments, does not require a line of sight, and is simple and easy to connect.
It was first developed by Philips and Sony. The current data exchange rate is approximately 424 kbps. [7]

- **Z-wave**:

Is a proprietary home automation protocol developed by Zensys in 1999. Later acquired by Sigma Designs.
Unlike most LAN protocols, Z-Wave operates in the sub-gigahertz band. This protocol is entirely proprietary. However, parts of the stack have been reverse-engineered to provide an open-source solution compatible with Z-Wave devices such as OpenZwave. The Z-Wave comes without encryption by default, so all Communications are sent in clear text, making it easier to intercept, collect, and decrypt messages.[8]

4.2. Broadband and long-range networks:

- **LPWAN (Low Power Wide Area Network):**

LoRaWAN is a low-power wide area network (LPWAN) for resource-constrained devices operating over long-distance networks. LoRaWAN is a MAC layer protocol based on the Long Range RF (LoRa) protocol and managed by the LoRa Alliance.[9]

- **NB-IoT (Narrowband IoT):**

Narrowband Internet of Things (NB-IoT) is one of the key technologies for fifth-generation (5G) mobile networks, and many industries have invested significant efforts in the development and research of NB-IoT systems. IoT.
NB-IoT is designed for low deployment/device costs, latency tolerance, long battery life (i.e. 10 years), large coverage areas, and a large number of low-speed devices that can send and receive small amounts of data with latency tolerance.[10]

4.3. The wired connection:

- **Ethernet** :

In the context of IoT, Ethernet connections are used to connect devices to local area networks (LANs). Although wired connections such as Ethernet provide fast and reliable communications, their use is often limited by the cost and impracticality of physical cables, especially for mobile

devices.

4.4. Specific communication technologies:

- **R-FID (Radio-Frequency Identification):**

RFID is known as radio frequency identification and is used to uniquely identify or track objects and objects. RFID technology is the basis of IoT and two terms are used: One is an RFID reader and the other is an RFID tag.

This technology works using radio waves transmitted by an RFID reader. An RFID tag is attached to an object that you want to track or identify, and whenever that object is near an RFID reader, that reader can read the signal using radio waves.[11]

5. IoT communication protocols:

5.1. MQTT:

Is also an application layer protocol that operates with limited resources and low bandwidth.

MQTT is used to provide communication between different devices in remote locations. It works like client-server communication. Used in MQTT protocol terms such as broker, publisher/subscriber, and subject.

The subject refers to the interest you need to register, i.e. the information you can expect from the publisher. Publishers publish posts based on a specified topic and send the posts to subscribers on that particular topic. The server or broker then receives messages from publishers and filters them to determine which subscribers to send them to. [4]

5.2. HTTP/HTTPS:

Hypertext Transfer Protocol (HTTP) is widely used to access and distribute data over the World Wide Web. It allows the transfer of various types of data such as text, hypertext, audio, and video. HTTP uses port 80 for communications. The main objective of the HTTPS (Hypertext Transfer Protocol Secure) protocol is to secure data exchange over an insecure network, such as the Internet. It creates a secure channel using a Secure Sockets Layer (SSL) certificate issued by a trusted certificate authority. Unlike HTTP, HTTPS uses port 443 for communications, providing a secure and encrypted connection between the client and server. [12]

5.3. CoAP :

Is an application layer protocol because all IoT devices have limited resources. Devices should be installed with integrated processors, limited RAM and ROM, limited bandwidth, and low power consumption. IoT devices are battery-powered and designed to last for long periods without needing to be recharged. Considering all these limitations of IoT devices, working on the web using

TCP/IP protocol is not that simple.

Therefore, a new protocol, COAP, was developed to work with restricted devices and transmit data from one location to another. COAP uses UDP instead of TCP/IP because UDP is a connectionless protocol and requires no computation to establish a connection. [4]

5.4. AMQP:

Is an application protocol used for communication between clients and servers. AMQP is an open standard protocol, so anyone can use it without paying any fees.

Previous protocols could not achieve interoperability by exchanging information from one place to another, but AMQP does. This way you can send messages and also receive confirmations. It is therefore very reliable. [13]

5.5. Bluetooth Low Energy (BLE):

BLE, known as Bluetooth Smart, is part of the Bluetooth v4.0 and now v4.2 stack and uses low data rates to rarely transmit small pieces of data. Is a global personal area network protocol designed to be Developed with significantly reduced power consumption per bit. It is a lightweight version of classic Bluetooth, intended for low-power and resource-constrained devices. [14]

5.6. Zigbee:

Is a short-range wireless communications standard for embedded devices and a mesh local area network (LAN) protocol originally developed for building control and automation. Offers significant advantages in complex systems, providing low power consumption, high security, robustness, and high scalability with a large number of nodes, making it advantageous for operating sensor networks and wireless control in IoT and M2M applications.[6]

5.7. LoRaWAN:

LoRaWAN is classified as a media access control (MAC) protocol designed to support large-scale public networks with a single operator. Additionally, instead of using narrowband transmission, data is sent using encrypted messages over different radio channels and transmission speeds.

LoRaWAN assumes that devices have different capabilities depending on their application requirements. Therefore, LoRaWAN defines three classes of end devices, all of which support duplex communication but have different power requirements and downlink delays.[15]

5.8. 6LoWPAN:

The 6LoWPAN protocol, essential for the Internet of Things (IoT), stands out for its ability to adapt IPv6 communications to low-power networks, such as those based on the IEEE 802.15.4 standard. It implements specific header compression to reduce transmission overhead and uses fragmentation to adapt IPv6 frames to the maximum size allowed by IEEE 802.15.4, typically 128 bytes.

6LoWPAN packets are structured according to different header types, including the 6LoWPAN header, distribution header, mesh header, and fragmentation header. The absence of the 6LoWPAN header indicates that the frames are not compatible and are therefore discarded. The distribution header provides compression of IPv6 headers and handles multicast, while the mesh header supports broadcast within the mesh network. Finally, the fragmentation header allows large IPv6 headers to be split into 128-byte fragments for efficient transmission over 6LoWPAN networks. [6]

6. IoT communication applications.

6.1. Intelligent Transportation System:

We live in an interconnected world where networks connect the physical world and the information world, improving every aspect of society, including the workplace. It's an incredibly difficult world to navigate in the world of IoT. A place where businesses can quickly manage their IT resources, control their environments and secure their networks as easily and modernly as they want. What makes transportation such a dynamic field is also what makes it one of the most technologically sophisticated and constantly evolving fields.

Transportation workers need more advanced communications to navigate a crowded world, stay safe on the roads, and manage railroads, airports, and ports.[16]

6.2. The health sector:

Imagine a truly modern healthcare system that everyone can access from the comfort of their own home. Systems using the Internet of Things make it possible to monitor health status, share data with healthcare providers, and alert others when needed. The home function allows you to monitor your patients 24 hours a day.

Built-in sensors track movements in your home and alert family members or emergency services if necessary. [17]

6.3. Smart home automation:

Most startups are now using IOT on a large scale and there is a lot of good research going on in this area. However, unfortunately, due to a lack of awareness, they send solutions that are familiar to them. Startups get a nice boost when it comes to innovation, but they're a little nervous about selling or building something innovative because they're used to the existing definition of a house-intelligent. IoT plays an important role in home automation.

As a simple example, artificial intelligence allows companies like Nest to power their Nest cameras, which recognize facial features and understand who is entering your room or home. Therefore, AI essentially enables better solutions for the end customer and improves the smart home experience through these different features. [4]

6.4. Smart cities:

With thousands of cameras and other live news sources, finding and following relevant events in any city is like looking for a needle in a haystack.

Adding a larger monitoring team doesn't necessarily help if the cognitive load becomes too high. Temporal situational awareness is essential for many urban surveillance tasks, but video processing networks and data centers can be unnecessarily burdened by unnecessary cameras.

Traditional decontextualized video recognition techniques are practically not applicable in highly variable and complex online situations, such as real urban environments. [18]

6.5. Smart farming:

IoT platforms can be used for smart precision agriculture. Some companies are leveraging cloud and mobile technologies to enable this for all types of agricultural, seed, and fertilizer, or R&D-based businesses.

IoT technology can be used for various forecasts such as weather, precipitation, wind direction, wind speed, solar radiation pressure, etc., and at the same time can also be used for some measurements such as soil temperature, soil moisture, pearl pH, soil EC, etc. This data is transferred to a Cloud platform. Then, once we receive the accessible information, this data is sent back to our device again. This device can also automate the agricultural sector. [19]

6.6. Environment and ecology:

The Internet of Things can play an important role in predicting the occurrence of natural disasters such as earthquakes, tsunamis, flash floods, thunderstorms, hailstorms, incessant rains, strong winds, and storm surges, as well as in the rapid evacuation of people from the area.

The deployment of sensor nodes in relevant areas and the mutual and internal communication between SHIP (Small, Partitioned, Intermittent, and Partitioned), a cluster-oriented network of IoT-enabled nodes, contributes to the rapid proliferation of nodes in isolated regions.

Disaster-affected areas should begin relief operations as soon as possible, and people should be rescued and brought to safety as quickly as possible. [20]

7. Problems encountered by IoT:

The Internet of Things plays an important role in life, but it suffers from many problems, including:

7.1. Security :

IoT security poses major challenges due to lack of standardization, hardware vulnerabilities, weak authentication, and concerns over unsecured data, cloud risks, and DDoS attacks.

These objects also suffer from numerous vulnerabilities, among these vulnerabilities including: [21]

- **Poor physical security** :

Insufficient physical security poses a major risk for IoT devices, which often operate in unmonitored environments. An adversary could easily gain physical access to these devices and take control, potentially revealing cryptographic schemes, duplicating firmware through a malicious node, or corrupting data. These attacks could cause physical damage to devices and compromise their integrity and operation. [21]

- **Insufficient energy recovery:**

This represents a major challenge for IoT devices, which often have limited energy resources and little ability to automatically recharge them. An attacker could exhaust available power by sending a large number of messages, legitimate or corrupt, thereby making devices inaccessible to legitimate processes or users. This attack would compromise device availability and operation, highlighting the importance of efficient energy management in IoT system security. [21]

- **Inadequate authentication:**

The insufficiency of authentication mechanisms represents a major danger in the context of IoT, due to specific limitations such as energy and computing power constraints, which complicate the implementation of authentication methods. As a result, an attacker could exploit weak authentication strategies to onboard spoofed malicious nodes or alter data integrity, compromising IoT devices and network exchanges. Furthermore, in such situations, the authentication keys exchanged and used are constantly at risk of loss, destruction, or corruption, especially when their storage or transmission is not secure. [21]

- **Incorrect encryption:**

Inadequate encryption is a major concern in IoT industries, especially in mission-critical systems such as power utilities, factories, and building automation. Encryption is crucial to secure data storage and transmission, allowing access only to authorized users. However, IoT resource constraints limit the robustness, effectiveness, and efficiency of encryption algorithms, making data vulnerable to interception or manipulation by attackers. Therefore, it is possible that an attacker could bypass the encryption methods used, thereby compromising data confidentiality or taking control of operations with minimal effort. [21]

- **Unnecessary open ports** :

Some IoT devices have unnecessary open ports running vulnerable services, allowing attackers to connect and exploit various security vulnerabilities. [21]

- **Insufficient access control:**

Insufficient access control is a major security gap in IoT systems, where robust credential management is crucial to protect devices and data from unauthorized access. Unfortunately, most IoT devices, along with their cloud management solutions, do not enforce sufficiently complex passwords or require users to change default credentials after installation. Additionally, many users have elevated permissions, making it easy for unauthorized access to devices, posing a threat to data and the integrity of the entire Internet. [21]

- **Inadequate patch management capabilities:**

Patch management capabilities are often inadequate in IoT systems, where operating system and firmware updates are essential to reduce the risk of attacks and improve performance. However, many manufacturers do not regularly maintain security patches, and available update mechanisms may lack automation or integrity guarantees, leaving them open to malicious tampering and large-scale vulnerabilities. [21]

- **Weak programming practices** :

Insufficient programming practices are a major concern in IoT, as many firmware are released with known vulnerabilities such as backdoors, use of root accounts as primary access, and lack of Secure Socket Layer (SSL). These flaws make it easy for an attacker to exploit known vulnerabilities to cause buffer overflows, corrupt data, or gain unauthorized access to the device. [21]

- **Insufficient audit mechanisms:**

Auditing mechanisms are often insufficient in many IoT devices because they lack detailed logging procedures, which helps hide IoT-generated malicious activities. [21]

7.2. Integration and interoperability:

Integration and interoperability in IoT pose major challenges due to divergent standards and protocols, leading to technical complexity and high costs. The diversity of IoT devices makes it difficult to seamlessly exchange data and collaborate among them, limiting their interoperability. Managing heterogeneous data generated by IoT devices requires suitable solutions for aggregation, storage, and analysis. Additionally, security and privacy are major concerns, as increased interoperability can increase the risks of vulnerabilities and attacks. Efforts are being made to develop common standards and best practices to promote interoperability and ensure the security of IoT systems.

7.3. Data confidentiality:

The issue of data privacy in IoT raises concerns due to the extensive and potentially intrusive collection of personal information. IoT devices collect data such as lifestyle habits and location, raising concerns about individual privacy. Security risks from IoT device vulnerabilities also increase fears of hacking and disclosure of unauthorized sensitive data. In addition, the diversity of suppliers and standards makes it difficult to implement consistent protection measures. Initiatives are underway to develop robust security regulations and standards to preserve user privacy while driving IoT development.

7.4. Data management:

Data management in IoT is complex due to the diversity, volume, and speed of data collection, as well as the fragmentation of data formats. Challenges include data aggregation, storage, analysis, and privacy. To meet these challenges, robust data infrastructures, effective communication protocols, and appropriate security measures are required, while complying with data protection regulations.

7.5. Scalability and infrastructure:

The scalability and infrastructure challenge in IoT is managing the rapid growth of connected devices and data generated, with often limited infrastructure. Connectivity and processing needs are constantly evolving, requiring flexible and scalable solutions. Cloud architectures and technologies such as edge computing are being explored to meet these requirements while ensuring system security and reliability.

7.6. Energy consumption :

The challenge of energy consumption in IoT is to ensure the efficient use of connected devices on limited energy sources. This requires minimizing power consumption while maintaining functionality and connectivity. Techniques such as intelligent power management and optimization of communication protocols are used. Additionally, adopting standards for the design of energy-efficient devices and exploring innovative solutions such as energy harvesting are important.

7.7. Reliability and maintenance:

Reliability and maintenance in IoT involve ensuring that devices continue to function in various environments. Hardware failures, communication interference, and software failures are major challenges to overcome. Large-scale maintenance can be complex and costly, requiring advanced diagnostic tools and efficient management processes. Proactive strategies such as designing robust devices and automating maintenance tasks are essential to improve operational efficiency and ensure the availability of IoT devices.

8. Attacks on IoT networks:

8.1. Physical attacks:

Physical attacks focus on hardware devices in the system :

- **Node tampering (Node Tampering):**

The attacker physically modifies a system node to obtain sensitive information such as encryption keys. To counter this attack, it is crucial to use physical intrusion detection mechanisms and integrity verification methods to ensure that nodes have not been tampered with. [22]

- **RF Interference on RFIDs :**

The attacker creates a denial of service by disrupting the radio frequency signals used for RFID communication. To counter this attack, robust frequency modulation and encryption techniques can be implemented to make RFIDs less vulnerable to interference. [22]

- **Node Jamming in WSNs:**

The attacker uses a jammer to disrupt wireless communication in wireless sensor networks (WSNs), thereby causing a denial of service. To counter this attack, interference detection and communication redundancy techniques can be employed to maintain connectivity despite disruptions. [22]

- **Malicious node injection Node Injection** :

The attacker physically injects malicious nodes into the network, thereby altering data and disrupting communication between legitimate nodes. To counter this attack, monitoring systems like MOVE can be used to verify the integrity of nodes and detect malicious behavior. Additionally, security mechanisms such as node authentication and data encryption can strengthen network security against such attacks. [22]

- **Physical Damage** :

The attacker physically damages the components of the IoT system, thereby leading to a denial of service attack. [22]

- **Social Engineering** :

The attacker physically interacts with users of the IoT system to manipulate them and obtain sensitive information to achieve his goals. [22]

- **Sleep deprivation attack (Sleep Deprivation attack)** :

The attacker aims to consume more power, which causes the system nodes to shut down. [22]

- **Malicious code injection:**

The attacker physically inserts malicious code into a node of the IoT system, allowing them to take full control of the system. [22]

8.2. Network attacks:

These attacks focus on the network infrastructures of the IoT system :

- **Attacks traffic analysis :**

The attacker intercepts and analyzes messages to obtain information about network operations. [22]

- **RFID theft (RFID Spoofing):**

An adversary spoofs RFID signals to capture information transmitted by an RFID tag. Spoofing attacks provide false but valid, data that is accepted by the system. [22]

- **RFID Cloning :**

The adversary copies data from one existing RFID tag to another, without including the original identifier. It can thus insert incorrect information or control the data passing through the cloned node. [22]

- **Unauthorized Access RFID:**

In the absence of proper authentication in RFID systems, an attacker can observe, modify, or delete

- **Chasm attack (Sinkhole attack):**

An adversary compromises a network node to send false routing information to its neighboring nodes, thereby attracting traffic. He can then modify or delete the data. A technique is proposed to identify these vulnerable nodes. [22]

- **Man in the Middle Attacks :**

The attacker intercepts the communication between two nodes to obtain sensitive information through eavesdropping. [22]

- **Denial of Service :**

The attacker overwhelms the network with excessive traffic, making services unavailable to legitimate users. [22]

- **Routing information attacks :**

The attacker complicates the operation of the network by spoofing, modifying, or sending routing information. This may result in packets being allowed or discarded, erroneous data being transferred, or network splitting. [22]

- **Sybil Attack :**

A malicious node assumes the identity of several other nodes and acts on their behalf, for example, in a voting system where a single node can cast multiple votes. [22]

8.3. Software attacks:

The attacker uses various methods, including :

- **Phishing attack :**

The attacker obtains confidential information such as usernames and passwords by spoofing emails and creating fraudulent websites. [22]

- **Viruses, worms, Trojan horses, spyware and adware :**

A malicious individual can compromise the system by injecting malicious code. These codes are often distributed via email attachments or downloads from the Internet. Worms have the particularity of reproducing without human intervention. To detect these threats, tools such as antiviruses, firewalls, and intrusion detection systems are used. [22]

- **Malicious Scripts :**

By injecting a malicious script, the attacker can access the system and compromise its security. [22]

- **Denial of Service :**

The attacker disrupts services by blocking users from the application layer, preventing them from accessing normally available services. [22]

8.4. Encryption attacks:

Encryption attacks aim to compromise encryption methods and obtain the private key, notably by:

- **Side channel attacks:**

The attacker exploits information emitted by encryption devices, such as strength and operation time, to deduce the key. Timing attacks are commonly used, exploiting timing variations in operations to reveal information about secret keys. [22]

- **Cryptanalysis attacks :**

The adversary seeks to obtain the encryption key by analyzing either the plain text or the cipher text. Methods include cipher text-only, known-plaintext, chosen-plaintext, and chosen-ciphertext attacks. [22]

- **Man in the Middle Attack :**

The attacker intercepts the communication between two users exchanging a key, allowing him to

obtain the secret key. [22]

9. Conclusion:

In conclusion, IoT is transforming our interaction with the world by connecting physical objects to collect and analyze data in real-time. Its varied applications, from health to smart cities, promise efficiency gains and improvements in quality of life. However, challenges in security, privacy, and interoperability must be overcome. With innovative solutions, IoT can become a pillar of digital society, making our environment more intelligent and sustainable.

Chapter 2:

State of the art of security mechanisms for IoT communication

1. Introduction:

In the first chapter, we provided an overview of the Internet of Things (IoT), including its various applications, the protocols it relies on, and the challenges it faces. Building on that foundation, this chapter will present an in-depth comparative study of several works that have successfully developed secure and effective systems for the transmission of data. We will examine the numerous security methods these works have employed to safeguard their IoT systems, analyzing the effectiveness and impact of these strategies in ensuring data security and system integrity.

2. Security methods:

To achieve the goal of data security, different security methods are used, including:

2.1. Cryptographic hashing algorithm:

- Zikang Ding et al, [23] propose a TXKLS algorithm, which is a lightweight key synchronization updating algorithm under a secure communication protocol for IoT devices LCPT. This protocol aims to strike a balance between security and performance, taking into account resource limitations. The security of this protocol is analyzed against common attacks, and formal verification is carried out using the Tamarin tool (this is software used to search for flaws in the security protocols of connected objects). This article evaluates the random quality and computational performance of the proposed algorithm, demonstrating its superiority over existing schemes.

  The results showed increased energy efficiency and high computational performance of TXKLS compared to two other algorithms TinyMT and XSadd. LCPT demonstrated robust computing performance while maintaining high security, LCPT's memory cost was evaluated and found to be low, making it applicable to most IoT devices, even those with extremely low resource limits.

2.2 . Blockchain:

- Mohamed Ali Kandi et al, [24] propose a decentralized key management protocol based on Blockchain and address the security challenges inherent in IoT device-to-device communication. By leveraging Blockchain technology, this solution provides resilience, scalability, and dynamic adaptability while mitigating the risks associated with centralized key management systems. This approach uses Blockchain technology to distribute and manage keys securely between network entities, eliminating the need for a centralized entity and reducing the risk of a single point of failure.

In the theoretical analysis, it was shown that the storage and computation costs on the nodes are propositional to the square root of the number of nodes in the network, while the communication remains constant. This shows that this solution maintains reasonable resource efficiency across nodes, even as the network grows.

In the evaluation of the performance of the Blockchain Participants, simulations were carried out which showed that the response time of the Blockchain participants depended on the number of Blockchain Participants in the system, as well as the processing operations of the Blockchain. transactions in the Blockchain. Plus the response time increases due to the increase in exchanges between them.

- Yongfeng Li [25], designed a secure interactive video surveillance data system to meet the growing security needs. By using a boost algorithm to process dynamic video frames, users can obtain more efficient video data. The cleaned video data is shared via the Blockchain platform. In the Blockchain network, the video is encrypted using the encryption algorithm, and the data is decrypted using the decryption algorithm, which ensures a level of security while sharing the video data. Experiments show that the sharing rate of video data and the degree of data confidentiality between different regions are relatively high, which can realize the sharing and encryption of the data transmission process and greatly improve the security performance of the interaction of video surveillance data.

  The system uses network cameras to capture images, which are then processed and secured via Blockchain technology. Digital fingerprint technology is used to guarantee the integrity of video data by identifying any potential alteration. Processed data can be shared with a remote monitoring center, ensuring secure and reliable monitoring between different regions.

  The experimental results confirmed the effectiveness of the video surveillance system, significantly reducing data loss and improving the security of video exchanges. Data protection measures were recommended, including the use of the HEVC/H.265 standard and the implementation of firewalls to strengthen network security.

2.3.Smart Contract :

- Bingcheng Jiang et al.[26], A publish/subscribe (P/S) system powered by Blockchain technology is proposed to enable secure one-way video sharing in the context of VEC. Additionally, presented an attribute-based encryption algorithm with static and dynamic attributes (ABE-SD) to achieve granular access control in mobile environments. Using Blockchain and permissioned smart contracts, recorded access policies and managed publish and subscribe events, enabling automatic user attestation and event tracking. The numerical

results show that the proposed ABE-SD scheme outperforms traditional centralized CP-ABE encryption methods in terms of encryption and decryption performance. Simulation experiments confirm that the proposed video-sharing system is secure and effective.

The results demonstrate the effectiveness of integrating Blockchain with the ABE-SD model to improve the performance of the video-sharing system. This approach significantly reduces publication and subscription times compared to conventional methods such as CP-ABE. In particular, the use of Blockchain facilitates efficient updating of dynamic vehicle attributes, which helps speed up transactions. Furthermore, smart contract analysis reveals that PBFT consensus generally offers shorter execution times than the Raft algorithm, suggesting optimal configurations for video sharing under the proposed system. Finally, optimizing batch sizes for publish and subscribe events helps keep smart contract execution times within a reasonable range, thereby improving the overall performance of the video-sharing system.

2.4 . Chaotic cryptography:

- Xin Huang et al. [27], propose an encryption framework to secure transmissions of video streams over ultrasonic communication channels through solid media, this method offers a reliable alternative where wired or wireless communications fail. The Chaotic Encryption Algorithm incorporates a 2D Arnold Cat Map and 1D Logistic Map to make the video stream secure and resistant to unauthorized access, this algorithm effectively scrambles the original video stream, thereby enhancing its security.

  The algorithm uses Chaotic encryption techniques, including Arnold Cat Map and Logistic Map, to transform video frames into seemingly random data. The Arnold Cat Map disrupts pixel position, while the Logistic Map substitutes pixel values, effectively blurring the original data. The results show a significant reduction in the correlation between pixels in the images before and after encryption, with a very low correlation coefficient in the encrypted images. Furthermore, the decryption process successfully restored the original videos from the encrypted images in full, confirming the effectiveness of the cryptosystem in securing the transmission of ultrasonic video data.
- Bachir Madani et al. [28], present a method for designing real-time network video communication systems while guaranteeing their security thanks to a chaos-based cryptosystem. This approach relies on a combination of hardware and software, implementing a TCP-IP protocol for video transmission. Synchronization is achieved by using FPGA PL-PS communication and implementing a custom algorithm for hardware and software at each end transmitting and receiving video from an OV7670 camera.

The hardware implementation results show that the receiver manages to successfully decrypt the original video when all parameters between the transmitter and receiver are correctly configured. This confirms the effectiveness of the proposed cryptosystem as well as the synchronization technique used. In addition, the results of the implementation of the hardware platform (transmitter/receiver) demonstrate low consumption in terms of FPGA resources. This suggests efficiency in the use of hardware resources, which is crucial for real-time applications such as video communication.

- Hayder Ayyed Naser et al. [29], present the development of a highly sensitive optoelectronic feedback chaotic communication system designed for secure data transmission. The system uses a chaotic signal generated by a solid-state laser, serving as a carrier for the data. A 10 Gb/s signal is encrypted and transmitted through free space optical (FSO) links and optical fibers.
  The communication system offers data transmission based on a chaotic CW laser, emitted at a wavelength of 1550 nm and directed towards an avalanche photodiode. This signal, amplified and returned to the laser source, is transmitted through free space and fiber optic channels. The receiver analyzes the incoming data using a low-pass filter and estimates the message based on the received intensity and the coupling coefficient.
  Experimental results demonstrate a transition from stable to chaotic oscillatory behavior by adjusting the feedback force. The system shows remarkable sensitivity to changes in this force, with a wide bandwidth allowing a variety of signal encoding methods. Furthermore, despite considerable transmission distances of up to 105 km, the system maintains a low bit error, illustrating its robustness and effectiveness in terms of data security.

2.5 . Video steganography:

- R.Umadevi [30], explored the area of secure communication, proposing the JA method in video steganography to improve the efficiency of data hiding while considering factors such as false positive rate and average PSNR. Experimental analyses focused on various video files with varying frame sizes and message overlay dimensions, using both irreversible and reversible techniques. Comparative evaluations with the PSO-IH and DCT-DH methodologies were also conducted, highlighting the effectiveness of the proposed JA method.
  This JA method obfuscates data using a video region vectorization technique, thereby reducing misclassifications of steganographed videos as original videos. The system evaluation is carried out on two sets of video and image data. The results show a significant reduction in false positive rate (FPR) of 28.18% compared to the PSO-IH method and 47.5% compared to the DCT-DH method. Additionally, the average signal-to-noise ratio (PSNR) is improved by

14.28% compared to PSO-IH and 8.5% compared to DCT-DH. These results indicate that the JA method provides better video quality and higher data privacy, making it a promising approach for secure video steganography communication.

- Karun Sharma et al.[31] propose an approach called PixelVerse, which is an innovative approach for secure file transfer, using video overlay to ensure data integrity and avoid compression. This article presents an in-depth evaluation of the PixelVerse methodology, including video encoding, compression prevention, data extraction, and import. Demonstrated the effectiveness of PixelVerse using three unique modes: Better Compression Resistance, Improved Compression Resistance, and Maximum Performance. Additionally, explored the applications and target markets of PixelVerse and provided real-world use cases. Research findings and evaluations indicate that PixelVerse proves to be an effective and reliable solution for secure file sharing and storage, as demonstrated by performance metrics and results. PixelVerse finds practical applications in various industries such as healthcare, finance, and legal services, where protecting data confidentiality and privacy during file storage and transfer is of paramount importance.

  PixelVerse works using video steganography, a method that hides data inside video files to ensure secure transfer. The process begins with preparing the video file, where each frame is processed individually to create an embedded source object. Then, the data to be transferred is divided into pieces and embedded in each video frame using the least important bits of the pixel values. Once the data is embedded, the frames are reassembled to form a new video file, which is encoded with a lossless codec to preserve quality.

2.6.Quantum cryptography :

- Jinze Li et al. [32], propose an identity authentication scheme for large-scale video surveillance systems, using quantum keys for enhanced security.

  The Identity and Quantum Key Key Generator uses quantum principles to produce secure keys. By integrating user identity and random numbers, it generates unique keys for authentication. This process ensures the security of large-scale video surveillance systems while allowing periodic updates to maintain security.

  The experimental results confirm the effectiveness of the identity-based key generator and quantum key, as well as the security authentication scheme. The analysis shows adequate key size, independence between the generated key and the seeds used, and linear efficiency despite increasing the number of users. Key matching shows significant improvement, with an increase

of approximately 70.16%, adapting the QKD network to the growing demand for keys for large-scale cameras.

2.7.Optical cryptography :

- Ahmed M. Elshamy et al. [33], present a video encryption technique and compare it to established methods such as Hénon's chaotic map and phase modulation schemes. As multimedia data volumes continue to increase in the commercial and military sectors, it is imperative to evaluate encryption techniques in terms of effectiveness, security, and noise immunity. The proposed video encryption technique aims to effectively address these concerns. This hybrid encryption algorithm combines Hénon's chaotic map and optical phase modulation to secure videos. It works by transforming each frame of the video into grayscale using Hénon's chaotic map, then applying optical phase modulation to make the images even more complex to decipher. Finally, the encrypted images are reintegrated in color to form an encrypted video. This process provides an effective method for protecting sensitive video data from unauthorized access.

  Simulation results show that the proposed encryption provides better performance in terms of low correlation with the original image, higher maximum and irregular deviation, and higher noise resistance compared to Hénon and optical phase modulation techniques. Additionally, it features increased sensitivity to key changes, while maintaining reasonable encryption time, making it a promising option for securing video data in various communication systems.

2.8 . Symmetric key cryptography:

- Hiren Kumar Deva Sarma [34], proposed a secure communication protocol designed for multimedia data transfer over mobile ad hoc networks (MANET). Using multi-path Ad hoc On-demand Distance Vector (AODV) as the basic routing protocol, the protocol emphasizes the security of the communication process. A new security scheme is introduced, using component coding elements, specifically YIQ, within a video stream to enhance security measures.

  The protocol provides a secure communication system for mobile ad hoc networks. Each node registers with a security server to obtain a keyring. Data is transferred securely using symmetric keys and digital signatures.

  The proposed protocol shows a significant improvement in throughput compared to SPREAD, recording higher values, attributed to its security measures. However, it consumes slightly more power due to the communication overhead of using security keys. The average path length is

reduced in the proposed protocol due to less packet loss, thus avoiding the need for retransmissions.

- Pradeep Pai T et al. [35], present a new video encryption approach focused on temporal and spatial efficiency. It proposes the use of region-swapping techniques followed by AES and DES encryption on MP4 video files. A time-selective encryption methodology is also introduced, using row and column swapping to efficiently mix video data. This method takes advantage of video compression, making the file playable in a distorted manner while providing enhanced security.

  The proposed method secures MPEG-4 files by encrypting sample data with RC and DES/AES permutation techniques. It uses a key file to store encryption information and facilitate the decryption process, ensuring data confidentiality during transport and storage. This process allows original files to be recovered from encrypted data, ensuring the integrity of audio and video data.

  Experimental results show that encryption time varies depending on file size and sample type, with longer times for larger file sizes and for combined encryption of audio and video. Both DES and AES have similar performance in terms of encryption times, although AES appears to be slightly faster for some files.

2.9 . Encryption based on entropy coding:

- Anoop BN et al.[36], presents a secure video transcoding method, which relies on the use of a correlation-preserving sorting algorithm to encrypt video content without requiring prior decryption.

  The study proposes an innovative method for secure video transcoding using a correlation-preserving sorting algorithm to encrypt video data without requiring prior decryption. This approach improves security by avoiding decryption of sensitive data while preserving compression performance. Video frames are selectively transmitted over a secure channel based on their correlation, while sorted frames are encoded according to MPEG2 and H.264 video coding standards. Experiment results show significant improvement in security and compression performance compared to traditional video transcoding methods, making it a promising solution to meet the increasing data security requirements in video processing.

2.10.Selective encryption :

- Kyungmin Go et al. [37], present an approach aimed at strengthening the security of battery-powered video transmission systems. proposed a secure video transmission framework that selectively encrypts video metadata and frames on a per-packet basis. This approach takes into

account factors such as available computing power and transmission distance to optimize encryption levels.

This secure video transmission framework offers selective encryption of video data, based on packet analysis to identify and encrypt meaningful data. Using information about available computing power and transmission distance optimizes the level of encryption. The restoration module ensures the correct reception of data by rearranging packets in their original order and handling possible losses.

The results show that the proposed framework consumes less energy than fully encrypted transmission while providing similar security. Additionally, it dynamically adapts to the available computing power, thereby optimizing battery usage. Finally, the framework provides robust protection against perceptual and cryptographic attacks thanks to its selective and adaptive encryption.

3. Comparative study:

Work	**Security method used**	**Data management**	**security**	**Energy management**	**Bandwidth**	**reliability**	**Allocation of computing capacities**
Zikang Ding and al [23] (2023)	Cryptographic hashing algorithm	do not require large data storage capacity	ensuring a high level of protection	do not require a large amount of energy	do not require high bandwidth	provide high reliability	require a high allocation of computing capacity
Mohamed Ali Kandi and al [24] (2020) + Yongfeng Li [25] (2023)	Blockchain	requires large data storage capacity	ensures high data security	energy consuming	require high bandwidth	ensure high reliability	require a high allocation of computing capacity

Bingcheng Jiang and al [26] (2023)	Smart contract	do not require large data storage capacity	ensure low data security	require a certain amount of energy	require high bandwidth to function properly	ensure high reliability	require a high allocation of computing capacity
Xin Huang and al [27] (2021) + Bachir Madani and al [28] (2023) + Hayder Ayyed Naser and al [29] (2023)	Chaotic cryptography	does not necessarily require large data storage capacity	provide a high level of security	require a certain amount of energy	require high bandwidth	provide a high level of reliability	require a high allocation of computing capacity
R.Umadevi [30] (2016) + Karun Sharma and al [31] (2023)	Steganography	requires sufficient storage capacity	provide a high level of data security	does not require high-energy	does not require high bandwidth	offering high reliability	require an allocation of computing capacity
Jinze Li and al [32] (2023)	Quantum cryptography	does not require large data storage capacity	provides high security	require a certain amount of energy	require relatively high bandwidth	offers high reliability	require an allocation of computing capacity

Ahmed M. Elshamy and al [33] (2017)	Cryptography optical	does not require large data storage capacity	provides high security	does not require high energy consumption	require high bandwidth	offers high reliability	require an allocation of computing capacity
Hiren Kumar Deva Sarma [34] (2018) + Pradeep Pai T and al [35] (2014)	Symmetric key cryptography	requires data storage capacity	ensures high data security	does not require high energy consumption	bandwidth can be influenced by the speed and amount of data exchanged	provide high reliability	can be influenced by the speed and quantity of data exchanged
Anoop BN and al [36] (2014)	Encryption based on entropy coding	does not necessarily require large data storage capacity	ensures high data security	do not require a lot of energy	require high bandwidth	guarantees high reliability	do not require allocation of computing capacity
Kyungmin Go and al [37] (2022)	Selective encryption	does not require large data storage capacity	ensures high data security	does not require high-energy	requires lower bandwidth	guarantees a high level of reliability	require an allocation of computing capacity

Table 2.1: Comparative study on related work

4. Discussion:

Hashing algorithms:

Hashing algorithms produce unique, fixed-size digital signatures from input data, ensuring data security by verifying its integrity, protecting passwords, and ensuring the authenticity of information. Robust algorithms like SHA-256 and SHA-512 are commonly used for this purpose. Although hashing calculations are not inherently energy-intensive, their use in specific contexts like cryptocurrency mining can require a lot of energy due to the intensive process of calculations

required. They may also require some computing capacity for complex operations or applications using hash tables, but they generally do not require high bandwidth because they work with local data.

Blockchain:

Blockchain requires large data storage capacity because each node in the network maintains a full copy of the Blockchain, creating a distributed and decentralized ledger that is highly secure and resistant to tampering. It ensures high data security by using advanced cryptographic mechanisms to ensure the integrity and confidentiality of stored information. Transactions are verified and validated by a network of distributed nodes, creating a tamper-resistant distributed ledger. However, Blockchain can be energy intensive, especially for cryptocurrencies like Bitcoin and Ethereum which use the proof of work mechanism. This mechanism requires a large amount of computing power, resulting in high energy consumption. Additionally, depending on the underlying protocols, Blockchain may require high bandwidth, particularly for networks using computationally intensive consensus mechanisms such as proof of work, which involve significant data exchange between network nodes. In contrast, other protocols like proof of stake may require less bandwidth because they do not involve the same level of computationally intensive activity.

Smart contracts:

Smart contracts are autonomous programs that automatically run on a Blockchain when certain conditions are met. They do not need a large data storage capacity, as they mainly contain programmed instructions. However, they can interact with data stored on the Blockchain or externally. Their security depends on their design and implementation, and they may require energy to operate, depending on the type of Blockchain used. Smart contracts are also prone to security risks and may require high bandwidth to function properly. Although smart contracts provide significant automation and security, their deployment requires careful planning in terms of security, energy consumption, and bandwidth.

Chaotic cryptography:

Chaotic cryptography focuses on the use of chaotic systems to perform cryptographic tasks, such as generating encryption keys and securely encrypting/decrypting data. It can offer a high level of privacy and security for data transmission, especially in commercial networks, providing an attractive alternative to traditional cryptography methods. Although chaotic cryptography may require some computational capacity to generate keys and encrypt data, it does not necessarily require large data storage capacity. It also makes it possible to quickly encrypt a large flow of information. Regarding bandwidth, the chaotic signal spectrum has a limited range of approximately 0kHz to 4kHz. This means that frequencies beyond 4kHz will not be masked or very

little by the chaotic signal. This can impact data transmission, particularly in commercial networks where higher bandwidth may be required to ensure fast and secure data transmission.

Steganography:

Steganography is a technique that hides data within other data, such as images or audio files, to make them indistinguishable. Although it requires some storage capacity to accommodate the hidden data, it does not require large storage capacity. Steganography provides a high level of data security by hiding information, making it difficult to detect and often requiring mathematical or statistical analysis. It is generally not power intensive and can even help save energy by reducing the bandwidth needed to transmit data over low-bandwidth networks. However, its implementation may require some computational capacity to efficiently run complex algorithms.

Quantum cryptography:

Quantum cryptography relies on the use of qubits to form encryption keys, ensuring high data security through quantum properties like superposition and entanglement. Unlike traditional cryptography, it does not require large data storage capacity. However, it requires some computational capacity to manipulate quantum states during key generation and transmission. In terms of performance, quantum cryptography can offer speed and energy efficiency advantages over classical cryptography. However, it may require a relatively high bandwidth to transmit encryption keys securely.

Optical cryptography

Optical cryptography exploits quantum properties to secure data, using individual photons to transmit encryption keys securely. It makes it possible to detect any attempt to intercept or measure the key, thus ensuring the integrity and confidentiality of the transmitted data. Unlike other cryptography methods, it does not require high power consumption. However, it may require high bandwidth for high-speed data transmission. The implementation of optical cryptography protocols may also require the allocation of computing capacity to ensure communications security.

Symmetric cryptography:

Symmetric cryptography uses the same key to encrypt and decrypt data. Although it requires storage capacity for secret keys, it is generally not very large. This method provides high security because it uses the same key for encryption and decryption operations, making the process faster and more efficient than asymmetric cryptography. However, security depends on the confidentiality of the key, which must be shared between the transmitter and receiver. If the key is compromised, data can be decrypted by unauthorized third parties. Symmetric cryptography does not necessarily require high power consumption. The bandwidth required can vary depending on several factors, such as the size of the data to be encrypted, the complexity of the algorithm used, and the speed of

data processing. Likewise, computing capabilities can be a critical factor, as it influences the speed of data processing, depending on the size of the data. data and the complexity of the algorithm used.

Entropy encryption:

Entropy encryption does not necessarily require large data storage capacity. It aims to ensure data security by making it more random and difficult to decrypt, using keys generated from entropy, which measures the degree of uncertainty of a random variable. However, data security primarily depends on the quality and proper implementation of encryption, using proven algorithms and following security best practices. Entropy can be generated from a variety of sources, some requiring little energy. Although some sources may require high bandwidth to collect entropy, others may not. When it comes to computational capabilities, entropy encryption methods generally do not require specific allocation.

Selective encryption

Selective encryption is a secure method for sharing sensitive information via cloud storage, without requiring large storage capacity. It ensures high data security while minimizing energy consumption. Although it generally requires less bandwidth than full encryption, specific requirements may vary by implementation. Proper allocation of computing resources is essential to ensure the performance, security, and energy efficiency of selective encryption in different applications.

5. Conclusion:

In this chapter, we analyzed a comparative study of various previous works that have developed secure and efficient communication systems. These systems are distinguished by their different encryption methods. This is because each system took a unique approach to encryption and data transmission, which led to a diversity of results. However, all of these systems play an important role in securing transmitted data.

Chapter 3:

Architecture based on Blockchain and smart contracts for IoT communication

1. Introduction:

In our previous chapter, we extensively examined numerous security methods designed to address the myriad challenges associated with the Internet of Things (IoT). Building upon this comprehensive analysis, we have resolved in this chapter to focus on one particular method as a definitive solution for ensuring security, integrity, authenticity, and confidentiality within IoT ecosystems. This selected method leverages the innovative application of Blockchain technology and smart contracts, complemented by the use of access codes. The integration of these technologies aims to establish a robust framework for secure and efficient communication within the IoT environment, thereby addressing the critical security concerns inherent to this rapidly evolving technological landscape.

2. IoT-Fog Computing Architecture for Data Security in an IoT Environment:

The proposed multi-layer architecture not only improves data security in an IoT environment but also optimizes system performance in terms of speed of response and ability to manage high volumes of data. Figure 3.1 represents our proposed architecture:

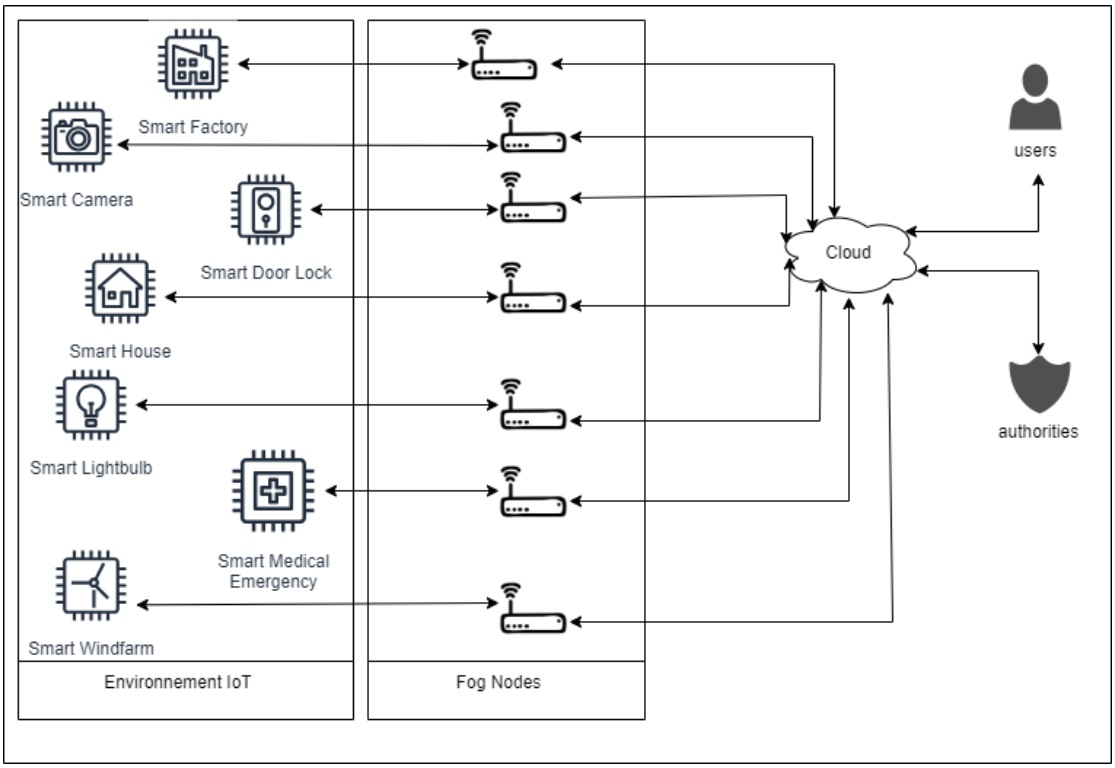

Figure 3.1: Architecture for communication in an IoT environment.

In an IoT environment integrating fog computing and cloud computing, communication plays a central role in ensuring connectivity between IoT devices, fog nodes, and cloud data centers. Data

is collected by IoT sensors and processed locally by fog nodes for quick decisions. Then, critical information is transmitted to cloud data centers for large-scale storage and analysis. Lightweight IoT communication protocols facilitate data transmission between devices and fog nodes, while network technologies like TCP/IP provide connectivity between fog nodes and cloud data centers. Security is paramount, with encryption and authentication mechanisms to protect data integrity and confidentiality.

2.1. Blockchain for integrity and immutability:

- **Secure storage**: Sensitive data is stored in the Cloud, while the Blockchain records cryptographic references or metadata linked to this data. This ensures that any changes to the original data are recorded and easily verifiable.
- **Immutability**: Once a transaction is recorded on the Blockchain, it cannot be altered without being detected by the entire network, thus strengthening the security and integrity of the data.

2.2. Fog Computing for Latency Reduction and Privacy Improvement:

- **Local processing**: Data is processed locally on fog computing nodes close to the IoT devices. This reduces latency by minimizing the time it takes to send data to remote data centers.
- **Increased Privacy**: By processing sensitive data locally, fog computing limits the amount of sensitive data passing through the network, thereby enhancing privacy.

2.3. Cloud Computing for Flexibility and Scalability:

- **Extensive processing capabilities**: The Cloud offers virtually unlimited processing and storage resources, enabling in-depth analysis of data collected by IoT devices.
- **Enhanced Security**: Cloud providers apply advanced security protocols to protect stored data from unauthorized access and external attacks.

2.4. Smart contract for Transparency and auditability

- **Automation and compliance:** Smart contracts automatically execute predefined rules and policies, ensuring compliance and rapid execution of security protocols without manual intervention.
- **Transparency and Auditability:** Actions carried out via smart contracts are recorded on the Blockchain, providing a transparent and auditable history of activities.

3. A secure model based on Blockchain in an IoT environment:

Through this model shown in Figure 3.2, we will present how Blockchain technology can be used to improve the security of data transmitted in an IoT environment.

More specifically, we will look at how to use Blockchain to send data such as a video to the user securely. Here is a figure that shows our model:

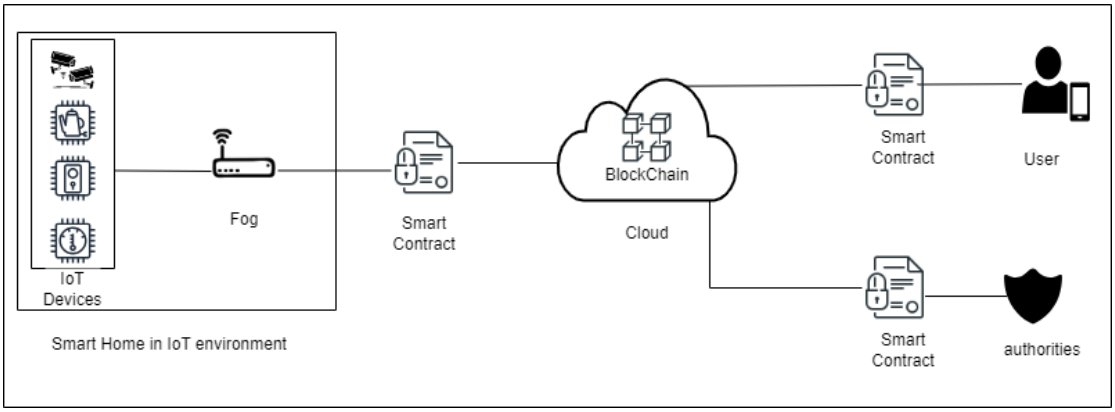

Figure 3.2: A Blockchain and Smart Contract Model for IoT

3.1. Explanation of the model:

Blockchain plays a role in securing data by guaranteeing its integrity and immutability. Sensitive data is securely stored in the Cloud, while Blockchain records cryptographic references or metadata linked to this data. Users then access the Cloud to view and interact with this data.

Secure Cloud Data Storage: Sensitive data is stored in the Cloud, where it benefits from advanced security measures such as encryption, access management, and threat monitoring. Cloud service providers typically offer secure storage solutions that meet the highest compliance and security standards.

Recording on the Blockchain: To ensure the integrity and authenticity of data stored in the Cloud, metadata or hashes of the data can be recorded on the Blockchain. This information serves as immutable proof of the existence and state of the data at a given time and helps detect any subsequent unauthorized changes.

Data Access via Cloud: Users can access data stored in the Cloud through user interfaces provided by the Cloud service provider. They can use web apps, mobile apps, or other tools to view, analyze, and interact with data based on their specific permissions and needs.

Blockchain-Enhanced Security: Blockchain enhances data security by providing an additional layer of verification and trust. By recording cryptographic references on the Blockchain, any attempt to modify or tamper with data in the Cloud can be detected and prevented, ensuring data integrity.

3.2. The structure of the Blockchain:

The BC is made up of blocks, each block is uniquely identified and timestamped to ensure the integrity and chronology of the recorded data. In figure 3.3 structure of Blockchain:

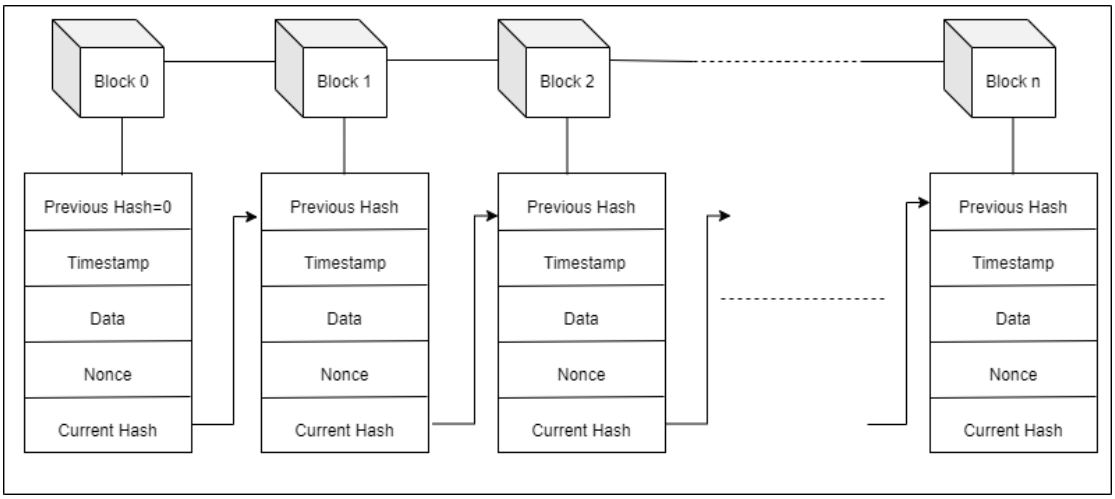

Figure 3.3: The structure of the Blockchain

Blockchain is based on a distributed network of nodes, each hosting a full copy of the Blockchain. When a transaction takes place, it is recorded in a block containing not only the transaction details but also a unique hash linking that block to its predecessor. Before being added to the chain, each block undergoes validation by network nodes to ensure transaction integrity. Once validated, the block is added to the chain, with its hash connecting the previous block, creating an immutable continuity of transactions. Consensus between nodes is ensured by different mechanisms, such as proof of work or proof of stake, which help ensure that all participants in the network agree on the state of the chain.

With each block addition, all nodes in the network are synchronized to reflect the latest version of the chain, ensuring an exact copy of the Blockchain for all participants. Blockchain security relies on cryptography, notably through the use of unique hashes to link blocks together. This makes any attempt to modify the data extremely difficult, as any alteration of one block would also affect subsequent blocks, which would be quickly detected by other nodes in the network.

3.3. Security protocol and smart contract:

Smart Contract is a standalone computer program that automatically executes the terms of a predefined contract when specified conditions are met. Using Blockchain technology, ensures transparency, security, and immutability of transactions, eliminating the need for intermediaries. Smart Contracts enable the automation of various activities such as financial transactions, digital asset management, and decentralized exchanges, paving the way for new business models and automation of business processes.

The smart contract lifecycle includes four phases:

-Creation of smart contracts (Creation): The parties involved negotiate the terms of the contract, and after several rounds of discussions, an agreement is drafted. This agreement is then transformed into a smart contract, written in computer languages, through design, implementation, and validation processes. This iterative process involves different parties such as stakeholders, lawyers, and software engineers.

-Deployment of smart contracts (Deployment): Validated smart contracts are deployed on Blockchain platforms. They become accessible to all parties via the Blockchain, and the digital assets of the parties involved are locked in corresponding digital wallets.

-Smart Contract Execution: Once deployed, smart contracts track contract clauses and evaluate specified conditions. When the conditions are met, the contract procedures are automatically executed. Transactions are validated by miners in the Blockchain, and updated states are stored on the Blockchain.

-Completion of smart contracts: After execution, the new states of the parties involved are updated, transactions are recorded on the Blockchain, and digital assets are transferred between the parties. At this point, the smart contract has completed its lifecycle.

Figure 3.4 shows the life cycle of smart contracts:

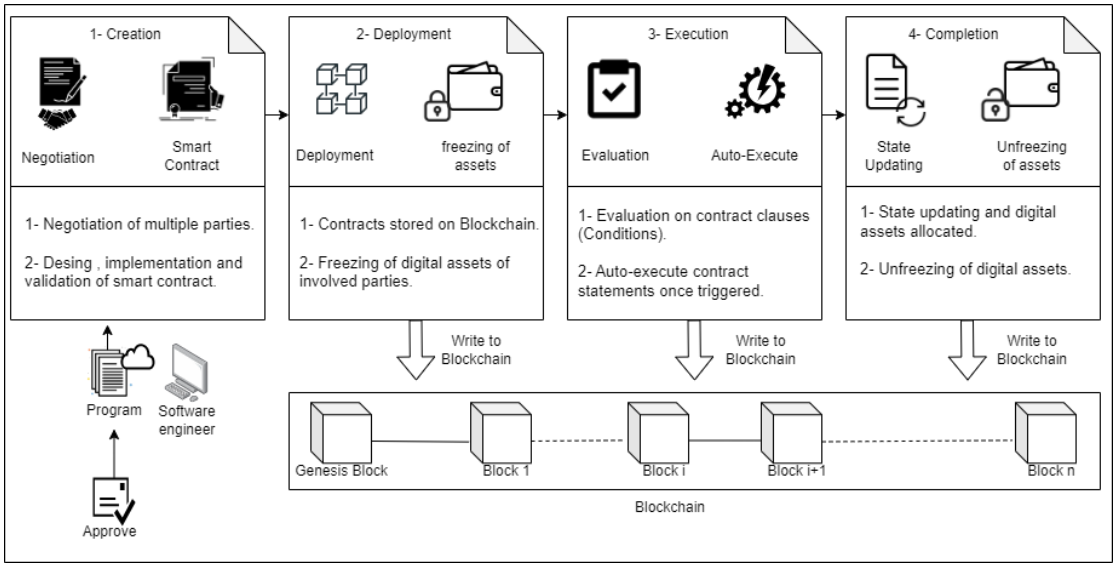

Figure 3.4: The life cycle of a smart contract

The following descriptions show the working principle and use of smart contracts to secure and automate critical aspects of data and device management in a complex IoT ecosystem. These contracts are considered in the context of an IoT environment integrating Blockchain, Cloud computing, and fog computing.

To implement an access token for IoT devices accessing the Cloud for data storage via a smart contract, we can design an Ethereum-based system using ERC-721 tokens, also known as Non-non-fungible tokens (NFTs). These tokens can represent unique access rights for each IoT device. Here are the detailed steps of this proposal:

Smart Contract (IoTAccessControl) to obtain an access token :

A smart contract named **IoTAccessControl** acts as a decentralized access manager for specific devices. It allows you to issue access tokens, validate access to a device with a token, revoke tokens, and transfer ownership of tokens between devices.

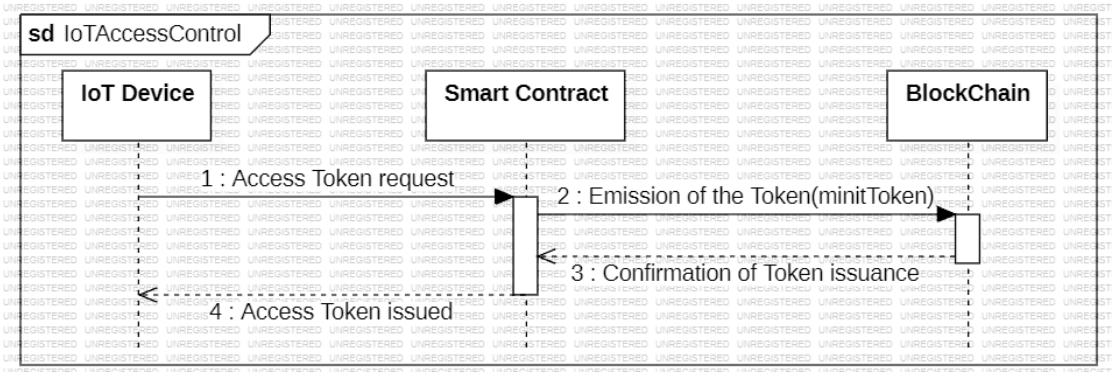

Figure 3.5: Smart contract sequence diagram (IoTAccessControl)

- The IoT device sends a request to the Smart Contract to obtain a new access token. This request can be triggered when a new device is added to the network or when an existing device wishes to update its access token.
- Upon receiving the request, the Smart Contract executes the **mintToken function**, which issues a new access token for the specified device. This token is recorded on the Blockchain, with the association of the device identifier and the owner.
- Once the issuance of the token is confirmed on the Blockchain, the Smart Contract returns the issued token to the IoT device, thus allowing it to access the network's functionalities.

Smart Contract for the Interaction of IoT Devices with the Cloud

This contract facilitates the interaction between IoT devices and the Cloud for data storage.

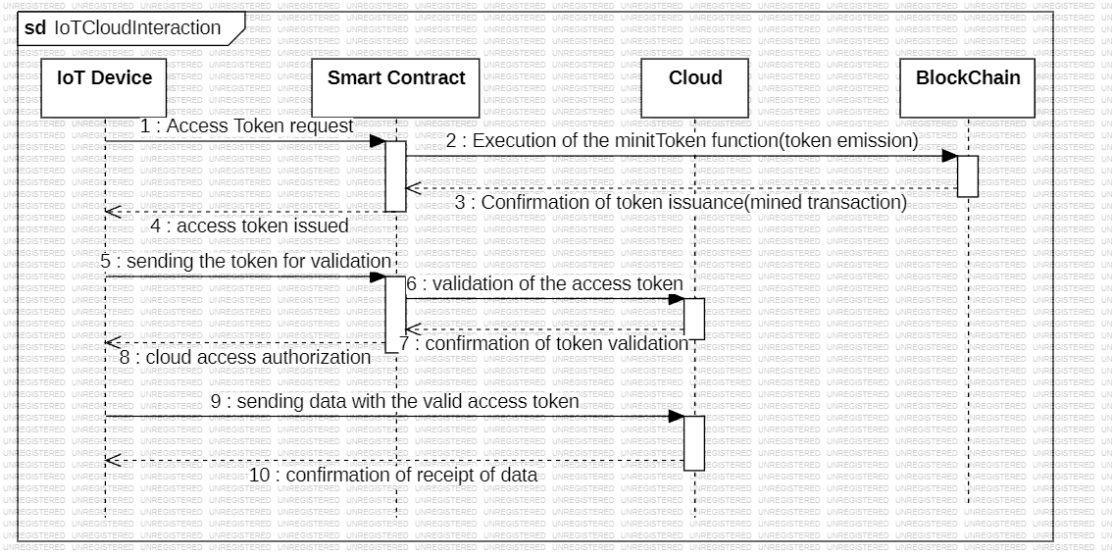

Figure 3.6: Smart contract sequence diagram (IoTCloudInteraction)

- The IoT device sends a request to the Smart Contract to obtain an access token, necessary to access data storage in the Cloud.
- The Smart Contract interacts with the Cloud to validate the access token requested by the IoT device. This validation ensures that only authorized devices can access the Cloud.
- Once the Cloud confirms the validation of the token, the Smart Contract authorizes access and returns this authorization to the IoT device. This authorization allows the device to send its data to the Cloud securely.
- The IoT device can then send its data to the Cloud, including the validated access token, ensuring that only data from authorized devices is stored in the Cloud.

Smart Contract for access authorization

This contract manages the authorization of access of IoT devices to the Cloud for data storage.

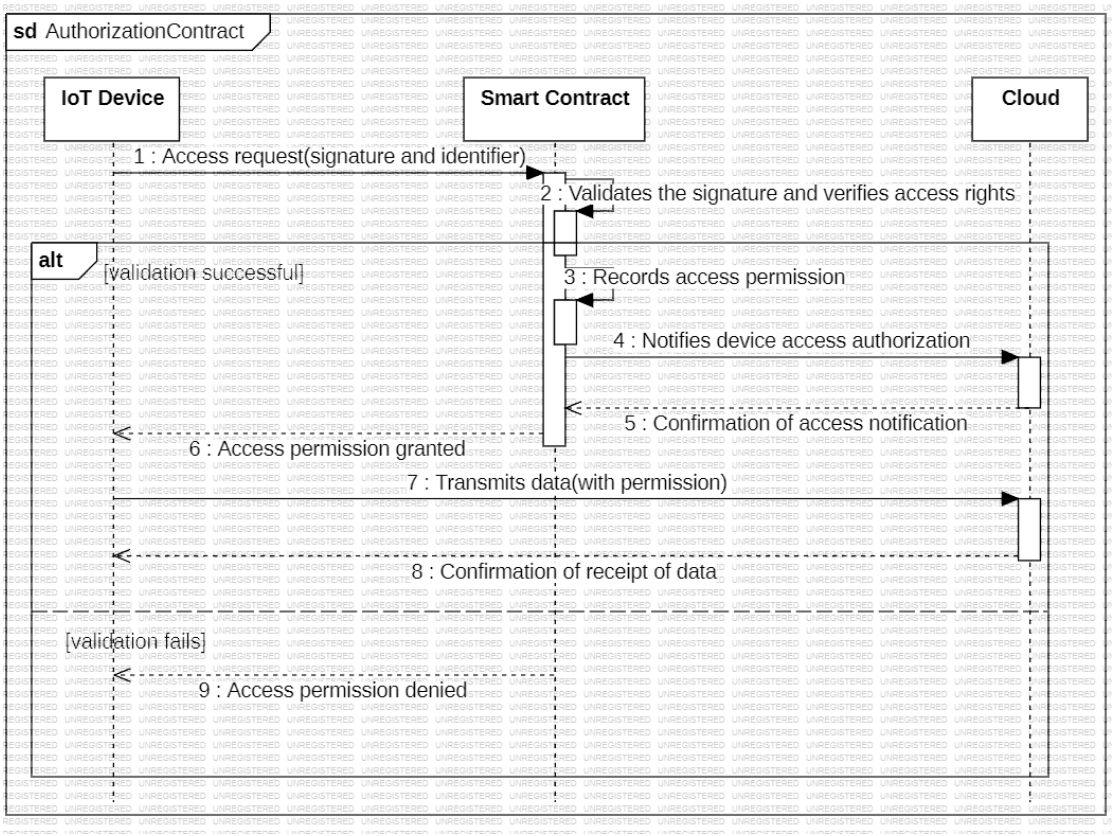

Figure 3.7: Smart contract sequence diagram (AuthorizationContract)

- The IoT device sends an access request to the Smart Contract to obtain authorization to transmit data to the Cloud.
- The Smart Contract validates the device's signature and verifies its access rights. This step ensures that only authorized devices can access the Cloud.
- If the validation is successful, the Smart Contract records the access and notifies the Cloud to allow access to the device. This allows the IoT device to send its data to the Cloud securely.
- The IoT device begins transmitting data to the Cloud, knowing that it has been authorized by the Smart Contract.

Smart Contract for data integrity verification

This contract guarantees the integrity of the stored data by verifying their conformity with the hashes recorded on the Blockchain.

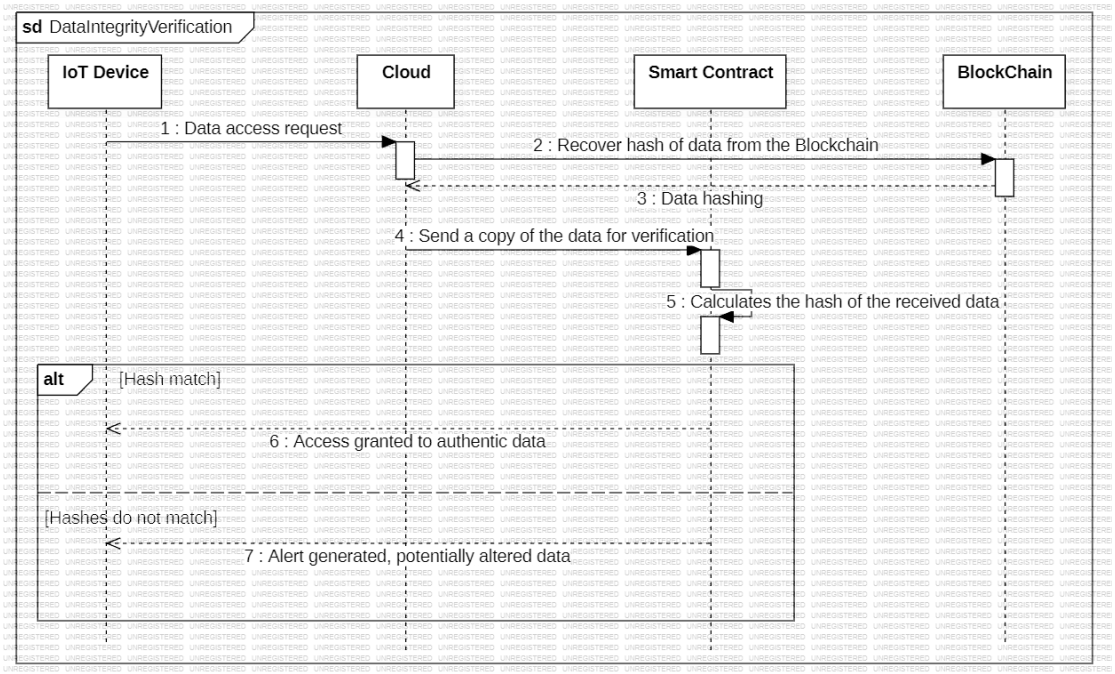

Figure 3.8: Smart contract sequence diagram (DataIntegrityVerification)

- The user requests access to data stored in the Cloud via a web interface.
- The Cloud retrieves the corresponding hash of the data from the Blockchain via a dedicated Smart Contract. This ensures the authenticity of the stored data.
- The Cloud sends a copy of the data to the Smart Contract for verification.
- The Smart Contract calculates the hash of the data received and compares it with the hash stored on the Blockchain. This comparison ensures that the data has not been altered since it was initially recorded.
- If the hashes match, access is granted to the user, allowing them to access the authentic data. Otherwise, an alert is generated, indicating possible data corruption.

Smart Contract for managing firmware updates

This contract automates the firmware update process for IoT devices, ensuring the security and authenticity of updates.

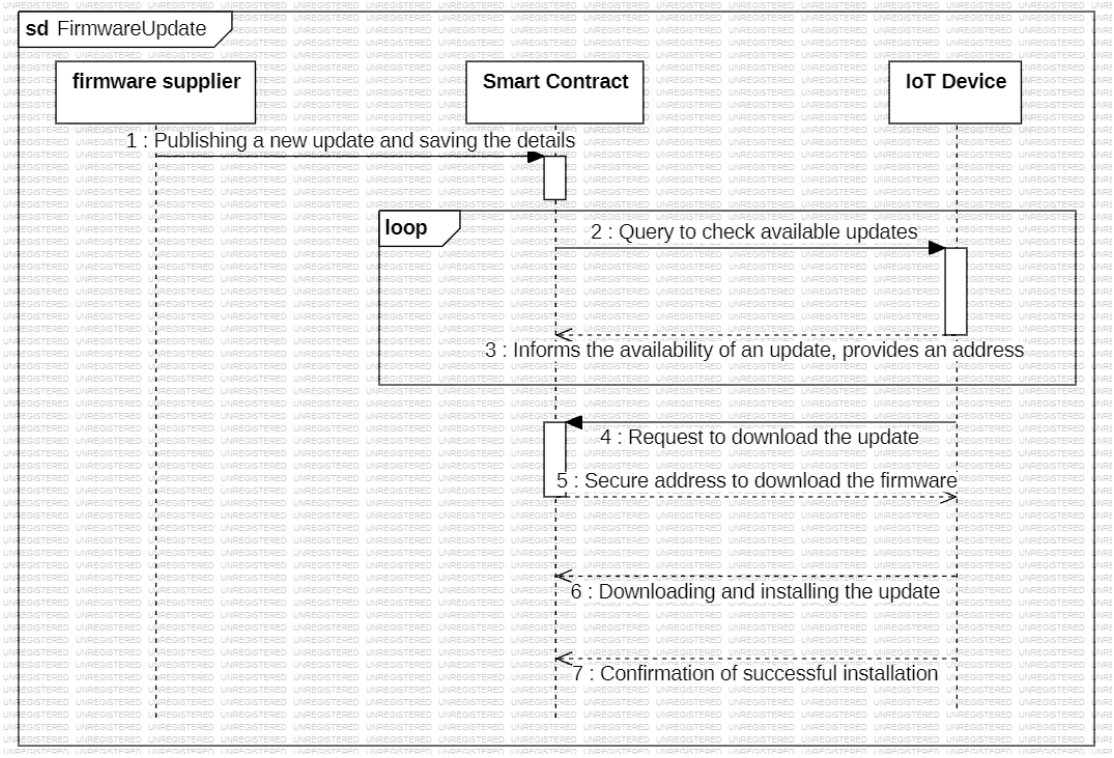

Figure 3.9: Smart contract sequence diagram (FirmwareUpdate)

firmware provider releases a new update and records the details in the Smart Contract on the Blockchain. This action ensures transparency and authenticity of updates.

- IoT devices periodically poll the Smart Contract to check for updates. This allows devices to stay up to date with the latest features and security patches.
- The Smart Contract informs devices of the availability of an update and provides a secure address to download the **firmware**. This approach ensures that updates are downloaded from trusted sources.
- IoT devices download and install the update, ensuring they are always equipped with the latest features and security patches.
- After installation, the devices send a confirmation of success to the Smart Contract, ensuring that the update process was completed correctly.

Smart contract between the user and the Cloud:

This contract grants the user several responsibilities, including access management, control, and security verification.

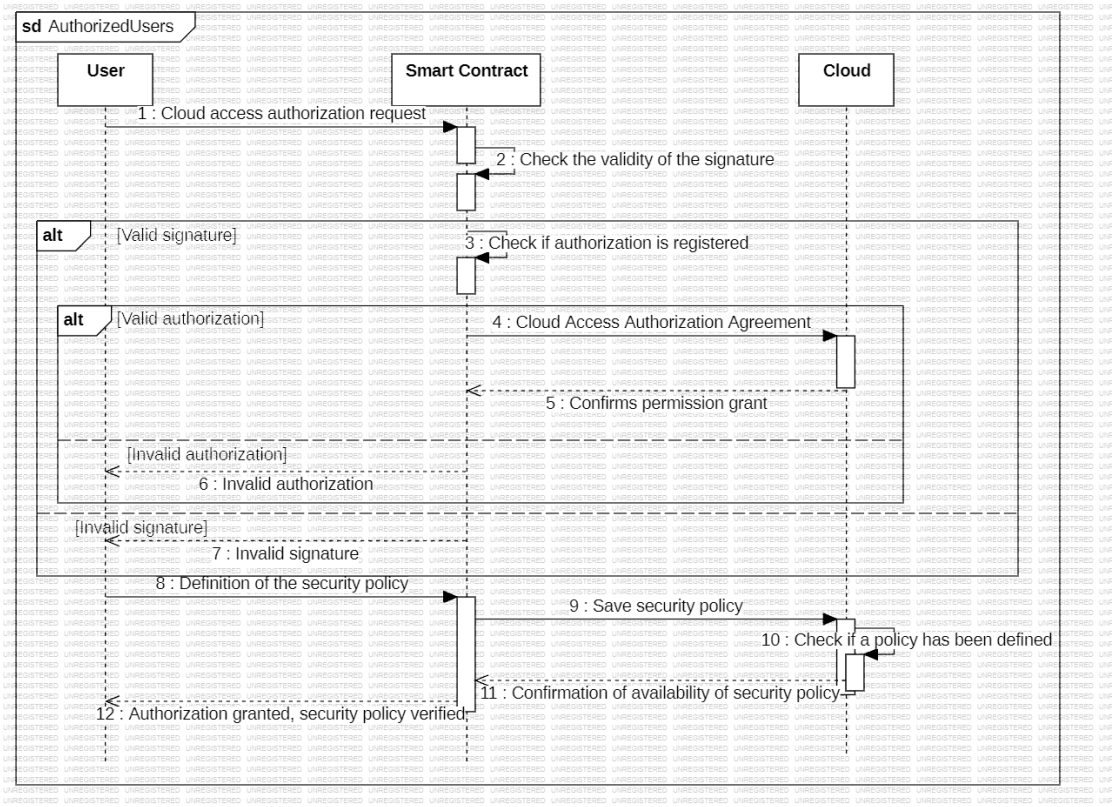

Figure 3.10: Smart contract sequence diagram (AuthorizedUsers)

-The user initiates the process by sending a Cloud access authorization request.

-The smart contract receives the authorization request and begins by verifying the validity of the accompanying digital signature; this signature must correspond to a valid authorization previously issued to the user.

-If the signature is valid and corresponds to an authorization recorded for the user, the smart contract recognizes that the authorization is legitimate.

-The smart contract grants Cloud access authorization to the user.

-The Cloud is legally faced with the obligation to check the availability of authorization for the devices that users seek to access.

-If the device is authorized, the Cloud grants the authorization and informs the smart contract of the decision.

-The user defines a security policy for their access to the Cloud, this step consists of the user establishing a set of rules, restrictions, and security mechanisms to control their access to Cloud resources.

-The Cloud saves the user-defined security policy to ensure compliance with specified rules.

-The Cloud checks if a security policy has been defined for the user, ensuring that all policies are in place for secure access.

-Once the security policy has been successfully verified, the Cloud grants permission to the user to access the data securely.

Smart contract between authorities and the Cloud:

This contract highlights access management, intervention management, and security verification

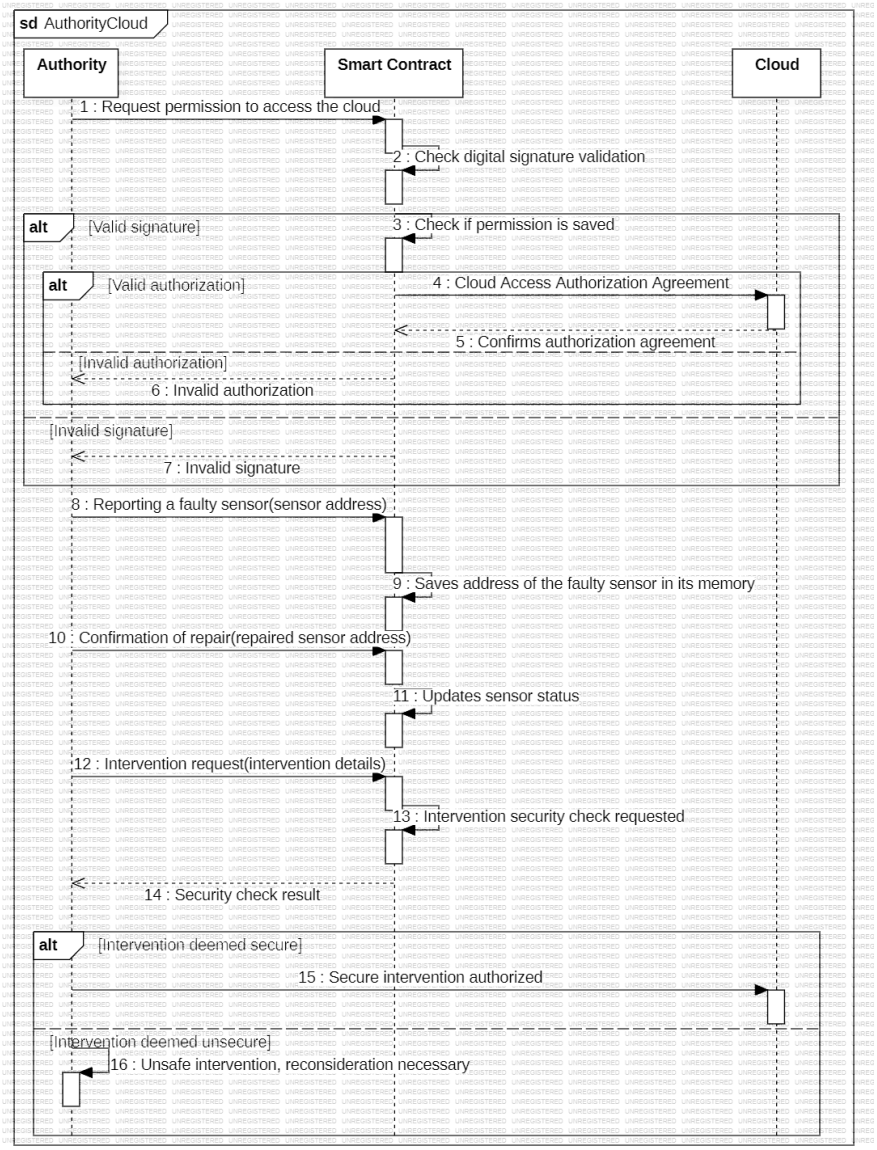

Figure 3.11: Smart contract sequence diagram (AuthorityCloud)

-The authorities send an authorization request to the smart contract to intervene in the system.

-When the smart contract receives an authorization request, it begins by checking the validity of the associated digital signature, this signature must correspond to an authorization previously

recorded for the authority concerned.

-If the signature is valid and corresponds to a registered authorization, the smart contract recognizes that the authorization is legitimate.

-The smart contract grants Cloud access authorization to the concerned authority.

-The authorities report a faulty sensor and this is done by the address of the faulty sensor.

-The smart contract receives the report and records the address of the faulty sensor in its memory.

-After repairing the sensor, the authority confirms the repair, and this is by recording the address of the repaired sensor.

-The smart contract updates the state of the sensor in its memory following repair confirmation.

-The authority requests smart contract intervention or details are provided about the requested intervention.

-The smart contract verifies the security of the requested intervention, and informs the authority of the result of the security check, if the intervention is deemed secure, the authority can then proceed with the intervention.

5. Conclusion:

In this section, we have presented an introduction to Blockchain, detailing its different types and how it works. We also proposed an architecture and a communication model using Blockchain and smart contracts to strengthen the security of data transmitted in a smart home.

Chapter 4:

Implementation and evaluation

1. Introduction:

In previous chapters, we explored the fundamental concepts of the Internet of Things (IoT), the associated security challenges, and the proposed architecture using Blockchain technology and smart contracts to improve the security of IoT communications. This chapter focuses on the practical implementation and evaluation of our secure communication system. We will detail the tools and technologies used for development, including the Solidity programming language and the Remix Ethereum IDE. We will then present the implementation of the different smart contracts developed to manage the access, storage, and verification of data from IoT devices.

After implementation, we will conduct a series of tests to evaluate the performance, security, and effectiveness of our system. We will also compare our solution to other existing approaches to highlight its advantages and disadvantages. Finally, we will discuss the results obtained and the prospects for improvement for future work.

2. Development tools and languages:

2.1. Solidity:

Solidity is a programming language specifically designed to create smart contracts on the Ethereum Blockchain. By providing syntax similar to JavaScript and C++, Solidity allows developers to define the logic and behavior of these contracts. These autonomous programs run on the Ethereum Blockchain and are responsible for the transparent, secure, and immutable execution of transactions and interactions within the decentralized ecosystem.[38]

2.2. Ethereum IDE Remix:

Remix IDE is an online integrated development environment specifically designed for creating smart contracts on the Ethereum Blockchain. It offers a range of tools for writing, testing, and deploying smart contracts efficiently. Compatible with languages like Solidity, Vyper, and Yul, Remix IDE allows the compilation, debugging, simulation, and deployment of smart contracts directly from the web browser.[39]

3. Implementation and creation of the system:

A smart contract (IoTAccessControl) to obtain an access token:

This smart contract functions as a decentralized access administrator for specific devices. It is

capable of generating access tokens, verifying them upon access requests, revoking them if necessary, and allowing transfer of ownership between devices.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";

contract IoTAccessControl is ERC721 {
    constructor() ERC721("IoTAccessControl", "IOTAC") {}   infinite gas 1713600 gas

    function mintToken(uint256 deviceId, address owner) external {   infinite gas
        _safeMint(owner, deviceId);
    }

    function validateAccess(uint256 deviceId, uint256 tokenId) external view returns (bool) {   infinite gas
        return ownerOf(deviceId) == msg.sender && ownerOf(tokenId) != address(0);
    }


    function revokeToken(uint256 tokenId) external {   infinite gas
        require(ownerOf(tokenId) == msg.sender, "Not token owner");
        _burn(tokenId);
    }

    function transferToken(uint256 tokenId, address newOwner) external {   infinite gas
        require(ownerOf(tokenId) == msg.sender, "Not token owner");
        safeTransferFrom(msg.sender, newOwner, tokenId);
    }
}
```

Figure 4.1: a smart contract to obtain an access token

-// **SPDX-License- Identifier: MIT**: Indicates the License of the contract.

- **pragma solidity >=0.7.0 < 0.9.0;** The version of Solidity needed to compile the contract.

-Import "@ openzeppelin / contracts /token/ERC721/ERC721.sol "; imports the OpenZeppelin ERC721 library, which is a standard implementation of ERC721, a standard for non-fungible tokens (NFTs) on Ethereum.

- **contract IoTAccessControl is ERC721**: declaration of the start of the **IoTAccessControl contract**, which inherits functionality from the ERC721 library.

- **constructor ( ) ERC721(" IoTAccessControl ", "IOTAC") {}** : This constructor is called during contract deployment. It calls the constructor of the ERC721 contract by passing it the parameters **IoTAccessControl** as the symbolic name and **IOTAC** as the symbol of the token.

- **mintToken (deviceId, owner):** This function is used to issue a new access **token for a specified device.** It takes as parameters the device identifier **deviceId** and the address of the owner **owner**. It uses the ERC721_safeMint function **to issue** a new **token** and assign it to the specified owner.

- **validateAccess (deviceId, token):** This function is used to check if a given token is valid for a specified device. It takes as parameters the device identifier **deviceId** and the **token** to be verified. It returns **true** if the owner of the **token** matches the message **sender** (the one that calls the function) and if the **token** exists.

- **revokeToken (tokenId):** This function revokes access by invalidating a specified **token**. It takes the identifier of the **token as a parameter token**. It verifies that the message **sender** is the owner

of the **token** before destroying it.

- transferToken (tokenId, newOwner): This function allows you to transfer ownership of a **token** to another user. It takes as parameters the identifier of the **token** to be transferred **tokenId** and the address of the new owner **new owner**. It verifies that the message **sender** is the owner of the **token** before transferring it to the new owner.

A smart contract for the interaction of IoT Devices with the Cloud

This contract facilitates the interaction between IoT devices and the Cloud for data storage.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

import "./IoTAccessControl.sol";

contract IoTCloudInteraction {
    IoTAccessControl public accessControl;

    constructor(address accessControlAddress) {    infinite gas 258200 gas
        accessControl = IoTAccessControl(accessControlAddress);
    }

    function requestToken(uint256 deviceId) external {    infinite gas
        accessControl.mintToken(deviceId, msg.sender);
    }

    function sendDataToCloud(uint256 deviceId, uint256 tokenId) view  external {    infinite gas
        require(accessControl.validateAccess(deviceId, tokenId), "Access denied");
        // Code pour envoyer les données au Cloud
    }
}
```

Figure 4.2: a smart contract for the interaction of IoT Devices with the Cloud

-// SPDX-License- Identifier: MIT: Indicates the License of the contract.

- pragma solidity >=0.7.0 < 0.9.0; The version of Solidity needed to compile the contract.

-import " ./IoTAccessControl.sol "; Imports the **IoTAccessControl.sol contract**, which is necessary to interact with the IoT Access Management contract.

- contract IoTCloudInteraction { IoTAccessControl public accessControl ;: IoTCloudInteraction contract declaration . It has a public accessControl variable of type IoTAccessControl, which is used to interact with the IoT access management contract.

-constructor (address accessControlAddress) { accessControl = IoTAccessControl (accessControlAddress);}: This constructor is called when deploying the IoTCloudInteraction contract . It takes an address as a parameter that corresponds to the address of the deployed IoTAccessControl contract. Then it initializes the accessControl variable with this address to enable interaction with the IoT access management contract.

- request token (deviceId): This function is used by an IoT device to request an access token from the **IoTAccessControl Smart Contract**. Once the request has been made, the **IoTAccessControl Smart Contract** issues a new **token** for the device.

- sendDataToCloud (deviceId, tokenId): This function is used by an IoT device to send data to the Cloud. Before sending the data, the device includes the device identifier **deviceId** as well as the corresponding access **token tokenId**. The Cloud then uses the **IoTAccessControl** Smart Contract to validate access before accepting the data for storage.

A smart contract for access authorization

This contract manages the authorization of access of IoT devices to the Cloud for data storage.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

contract AuthorizationContract {
    mapping(address => bool) public authorizedDevices;

    function requestAccess() external {    24522 gas
        authorizedDevices[msg.sender] = true;
        // Notification au Cloud pour autoriser l'accès
    }

    function sendDataToCloud() view external {    2675 gas
        require(authorizedDevices[msg.sender], "Unauthorized device");
        // Code pour envoyer les données au Cloud
    }
}
```

Figure 4.3: a smart contract for access authorization

-// **SPDX-License- Identifier: MIT**: Indicates the License of the contract.

- pragma solidity >=0.7.0 < 0.9.0; The version of Solidity needed to compile the contract.

- contract AuthorizationContract {: Declaration of the start of the AuthorizationContract contract.

- mapping (address => bool) public authorizedDevices; Declaration of an **authorizedDevices** state variable, which is a mapping **that associates each** device address with a boolean indicating whether it is authorized or not.

- requestAccess ( ): This function is used by an IoT device to request access to the Cloud. Once the request is made, the Smart Contract registers the device as authorized and notifies the Cloud to allow access.

- sendDataToCloud ( ): This function is used by an IoT device to send data to the Cloud once access has been granted. Before sending the data, the device is verified as authorized by the Smart Contract.

A smart contract for data integrity verification

This contract guarantees the integrity of the stored data by verifying their conformity with the hashes recorded on the Blockchain.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

contract DataIntegrityVerification {
    mapping(bytes32 => bool) public storedHashes;

    function storeHash(bytes32 hash) external {    24729 gas
        storedHashes[hash] = true;
    }

    function verifyData(bytes calldata data, bytes32 hash) external view returns (bool) {    infinite gas
        return storedHashes[hash] && keccak256(data) == hash;
    }
}
```

Figure 4.4: A smart contract for data integrity verification

-// **SPDX-License- Identifier: MIT**: Indicates the License of the contract.

- **pragma solidity >=0.7.0 < 0.9.0;** The version of Solidity needed to compile the contract.

- **contract DataIntegrityVerification {**: declaration of the start of the **DataIntegrityVerification contract.**

- **mapping (bytes32 => bool) public storedHashes**; declaration of a **storedHashes** state variable, which is a mapping **that associates each** hash **with** a boolean indicating whether it is stored or not.

- **stores (hash):** This function allows you to store the hash of the data on the Blockchain.

- **verify data (data, hash):** This function is used to verify the integrity of data. It takes as parameters the data itself **data** and the expected **hash hash**. It compares the calculated hash of the data with the hash stored on the Blockchain to determine if the data is authentic.

A smart contract for managing firmware updates:

This contract automates **the firmware** update process for IoT devices, ensuring the security and authenticity of updates. Here is how it works:

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.7.0 <0.9.0;

contract FirmwareUpdate {
    mapping(uint256 => string) public firmwareUpdates;
    mapping(uint256 => bool) public updateCompleted;

    function publishUpdate(uint256 version, string calldata firmwareHash) external {    infinite gas
        firmwareUpdates[version] = firmwareHash;
    }

    function checkForUpdate(uint256 deviceVersion) external view returns (bool) {    5211 gas
        return bytes(firmwareUpdates[deviceVersion]).length > 0 && !updateCompleted[deviceVersion];
    }

    function updateDevice(uint256 deviceVersion) external {    27085 gas
        require(bytes(firmwareUpdates[deviceVersion]).length > 0, "No update available");
        // Téléchargement et installation de la mise à jour
        updateCompleted[deviceVersion] = true;
    }
}
```

Figure 4.5: A smart contract for managing firmware updates.

-// **SPDX-License- Identifier: MIT**: Indicates the License of the contract.

- **pragma solidity >=0.7.0 < 0.9.0;** The version of Solidity needed to compile the contract.

- contract FirmwareUpdate {: Declaration of the start of the **FirmwareUpdate contract.**

- mapping (uint256 => string) public firmwareUpdates; Declaration of a **firmwareUpdates** state variable, which is a mapping **that associates each** firmware version with its hash.

- mapping (uint256 => bool) public updateCompleted; Declaration of an updateCompleted state variable, which is a mapping that indicates whether the update for a certain version has been completed or not.

- publishUpdate (version, firmwareHash): This function is used by the **firmware provider** to publish a new update. It records the update details, including the version number and firmware hash**,** in the Smart Contract.

- checkForUpdate (device version): This function is used by IoT devices to check the availability of updates. It returns **true** if an update is available for the specified version of the device and it has not yet been installed.

- updateDevice (device version): This function is used by IoT devices to download and install an available **firmware** update. Once the update is successfully installed, the device sends a confirmation to the Smart Contract.

A smart contract between the User and the Cloud:

This program is a Smart Contract in Solidity which implements a communication system between the user and the Cloud. This smart contract manages the users authorized to interact with a device. It allows the contract owner to add authorized users and restricts access to certain functions only to the owner. It uses the "ECDSA" library to verify digital signatures and ensures that only authorized users can access or interact with the device.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

library ECDSA {
    function recover(bytes32 hash, bytes memory signature) internal pure returns (address) {
        require(signature.length == 65, "Invalid signature length");

        bytes32 r;
        bytes32 s;
        uint8 v;

        assembly {
            // Signature format is [v, r, s]
            r := mload(add(signature, 32))
            s := mload(add(signature, 64))
            v := byte(0, mload(add(signature, 96)))
        }

        if (v < 27) {
            v += 27;
        }

        require(v == 27 || v == 28, "Invalid signature 'v' value");

        return ecrecover(hash, v, r, s);
    }

    function toEthSignedMessageHash(bytes32 hash) internal pure returns (bytes32) {
        return keccak256(abi.encodePacked("\x19Ethereum Signed Message:\n32", hash));
    }
}

contract AuthorizedUsers{
    //Structure pour representer un utilisateur
    struct User {
        bool exists;
        uint256 timestamp;
        // Autres champs si nécessaire
    }
    //Mapping pour stocker les utilisateurs autorises
    mapping (address => User) public users;
    //Adresse du proprietaire du contrat
    address public owner;
    //Modification pour restreindre l'acces aux fonctions du proprietaire seulement
    modifier onlyOwner(){
        require(msg.sender ==owner, "Only owner can call this function");
        _;
    }
    //Evenement declenche lorsqu'un nouvel utilisateur est ajoute
    event UserAdded(address indexed user);
    //Constructeur du contrat
    constructor(){
        owner = msg.sender;
    }
    // Fonction pour ajouter un utilisateur autorisé
    function addUser(address _userAddress) external onlyOwner {
        // Vérifier si l'utilisateur existe déjà
        require(!users[_userAddress].exists, "User already exists");

        // Ajouter l'utilisateur
        users[_userAddress] = User(true, block.timestamp);

        // Emettre un événement pour signaler l'ajout de l'utilisateur
        emit UserAdded(_userAddress);
    }
    //Fonction pour verifier la signature numerique
    function verifySignature(bytes memory _signature, bytes32 _messageHash) internal pure returns (address){
        bytes32 ethSignedMessageHash = ECDSA.toEthSignedMessageHash(_messageHash);
        return ECDSA.recover(ethSignedMessageHash, _signature);

    }
    //Fonction pour autoriser l'acces a un dispositif
    function grantAccess(bytes memory _signature, bytes32 _messageHash) view  external {
        address user = verifySignature(_signature, _messageHash);
        require(users[user].exists, "User not authorized");
        //Logique pour autoriser l'acces au dispositif
    }
    //Fonction pour interagir avec le dispositif
    function interactWithDevice(bytes memory _signature, bytes32 _messageHash) view  external {
        address user = verifySignature(_signature, _messageHash);
        require(users[user].exists, "User not authorized");
        //Logique pour interagir avec le dispositif
```

Figure 4.6: The smart contract (User-Cloud)

-// **SPDX-License- Identifier: MIT**: Indicates the License of the contract.

- pragma solidity >=0.4.22 < 0.9.0; The version of Solidity needed to compile the contract. Implement a Solidity library named **ECDSA** which provides functionality for working with digital signatures according to Ethereum specifications, it contains two functions:

- Function **recover** takes a message hash and signature as input, then extracts the signature components (r, s, and v) from the raw signature data. It then uses Ethereum's **recovery** function to retrieve and return the address associated with the signature.

- **The toEthSignedMessageHash** function takes a message hash as input and transforms it into a signed Ethereum message hash by applying a specific prefix conforming to the Ethereum standard.

-This contract introduces a **User structure** intended to represent users in the system. This structure includes two attributes:

exists: a boolean notifying the presence of the user.

Timestamp: An integer indicating when the user was added or modified.

This provides a methodical organization for storing data relating to system users.

UserCloud contract builder automatically initializes the **owner variable** with the address of the

person deploying the contract, thus establishing the owner as soon as the contract is deployed on the Blockchain.

addUser function allows the contract owner to add an authorized user by specifying their address. It first checks if the user does not already exist. If this is the case, it adds it to the **mapping users**, setting the **exists field** to **true** and saving the current **timestamp**. Finally, a **UserAdded event** is emitted to signal the addition of the user.

verifySignature function takes as input a digital signature and a message hash. It converts the message hash into an Ethereum-compliant format. Then, it uses the **ECDSA** library to extract the address associated with the signature using the message hash and the signature. Finally, it returns this address for verification.

grantAccess feature allows a user to request access to a device by providing a digital signature and message hash. It verifies the validity of the signature using the **verifySignature function**. Then, it checks if the user associated with the retrieved address is authorized by consulting the **mapping users.** If the user is authorized, access to the device is allowed.

interactWithDevice feature allows an authorized user to interact with a device by providing a digital signature and message hash. It verifies the validity of the signature using the **verifySignature function**. Then, it checks if the user associated with the retrieved address is authorized by consulting the **mapping users**. If the user is authorized, interaction with the device is permitted.

A smart contract between Authorities and the Cloud:

This program is a Smart Contract in Solidity which implements a communication system between authorities and the Cloud. The ` AuthorityCloud` contract manages the authorization and revocation of authorities which can report and repair faulty sensors, as well as request interventions for users. It ensures that only authorized and non-failing authorities can interact with the system. The contract maintains a transparent record of critical events for the effective management of sensors and interventions.

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

contract AuthorityCloud {
    address public cloud; // Adresse du cloud
    mapping(address => bool) public authorizedAuthorities; // Mapping des autorités autorisées
    mapping(address => bool) public malfunctioningSensors; // Mapping des capteurs défaillants

    event AuthorityAuthorized(address authority); // Événement émis lorsqu'une autorité est autorisée
    event SensorMalfunctioning(address sensor); // Événement émis lorsqu'un capteur est défectueux
    event SensorRepaired(address sensor); // Événement émis lorsqu'un capteur est réparé
    event InterventionRequested(address user, bytes32 action); // Événement émis lorsqu'une intervention est demandée

    constructor() {
        cloud = msg.sender; // Initialise l'adresse du cloud avec l'adresse du déploiement du contrat
    }

    modifier onlyCloud() {
        require(msg.sender == cloud, "Caller is not the cloud"); // Vérifie que l'appelant est le cloud
        _;
    }

    modifier onlyAuthorizedAuthority() {
        require(authorizedAuthorities[msg.sender], "Caller is not an authorized authority"); // Vérifie que l'appelant est une autorité autorisée
        _;
    }

    // Fonction pour autoriser une autorité
    function authorizeAuthority(address _authority) external onlyCloud {
        authorizedAuthorities[_authority] = true; // Autorise l'autorité en définissant sa valeur à true dans le mapping
        emit AuthorityAuthorized(_authority); // Émet l'événement d'autorisation d'autorité
    }

    // Fonction pour révoquer l'autorisation d'une autorité
    function revokeAuthority(address _authority) external onlyCloud {
        authorizedAuthorities[_authority] = false; // Révoque l'autorisation de l'autorité en définissant sa valeur à false dans le mapping
    }

    // Fonction pour signaler un capteur défectueux
    function reportSensorMalfunction(address _sensor) external onlyAuthorizedAuthority {
        malfunctioningSensors[_sensor] = true; // Marque le capteur comme défectueux en définissant sa valeur à true dans le mapping
        emit SensorMalfunctioning(_sensor); // Émet l'événement de capteur défectueux
    }

    // Fonction pour réparer un capteur défectueux
    function repairSensor(address _sensor) external onlyAuthorizedAuthority {
        malfunctioningSensors[_sensor] = false; // Répare le capteur en définissant sa valeur à false dans le mapping
        emit SensorRepaired(_sensor); // Émet l'événement de capteur réparé
    }

    // Fonction pour demander une intervention
    function requestIntervention(address _user, bytes32 _action) external onlyAuthorizedAuthority {
        require(!malfunctioningSensors[msg.sender], "Cannot intervene due to malfunctioning sensor"); // Vérifie que le capteur de l'appelant n'est pas défectueux
        emit InterventionRequested(_user, _action); // Émet l'événement de demande d'intervention
        // Logique de traitement de la demande d'intervention
    }
}
```

Figure 4.7: The smart contract (Authority -Cloud)

-// **SPDX-License- Identifier: MIT**: Indicates the License of the contract.

- pragma solidity >=0.4.22 < 0.9.0; The version of Solidity needed to compile the contract.

- public cloud address; create a public **Cloud** variable of type **address** which is used to store the address of the Cloud associated with the contract.

-Mapping **authorizedAuthorities** associates authority addresses with Boolean values. It is used to track which authorities are authorized to interact with the contract.

-Mapping **malfunctioningSensors** associates sensor addresses with Boolean values. It is used to track which sensors are faulty.

-Event declarations define four different events that can be emitted by the contract. They are used to report authority authorization, sensor failures, repairs, and service requests with associated details.

-The manufacturer initializes the **Cloud address** with the address of the person deploying the contract. This makes it possible to define the initial owner of the contract as soon as it is deployed on the Blockchain.

onlyCloud modifier checks if the caller of a function is the address of the **Cloud**. If it is not the Cloud, an exception is thrown with a specific error message. This modifier is used to restrict access to certain functions only to the Cloud, thus ensuring security and control of sensitive contract

operations.

-The **onlyAuthorizedAuthority modifier** checks whether the caller of a function is an authorized authority. If it is not, an exception is thrown with a specific error message. This modifier is used to limit access to certain functions only to authorized authorities, thereby increasing security and control over sensitive contract operations.

authorizeAuthority function, accessible only through the Cloud, allows you to authorize a specified authority by setting its value to **true** in the **mapping authorizedAuthorities**. It also emits an event to signal the authority's authorization, thereby providing a trace of authorization changes in the contract.

revokeAuthority function, accessible only through the Cloud, revokes the authorization of specified authority by setting its value to **false** in the **mapping authorizedAuthorities**. This means that the authority is no longer authorized to carry out operations in the contract.

reportSensorMalfunction function, accessible only by authorized authorities, allows you to report that a specific sensor is defective by setting its value to **true** in the **mapping malfunctioningSensors**. It also emits an event to notify relevant parties of the sensor failure, ensuring efficient management of failures in the system.

repairSensor function accessible only by authorized authorities, marks a specific sensor as repaired by setting its value to **false** in the **mapping malfunctioningSensors**. It also emits an event to notify relevant parties of the sensor repair, which helps track sensor repairs within the contract and ensures proper system operation.

request intervention function allows an authorized authority to request intervention for a specific user with a particular action. Before issuing the request, it checks that the authority's sensor is not defective. It then emits an event to signal the intervention request and can execute logic to process the request.

4. Execution of smart contracts:

Contract execution (IoTAccessControl):

Here are the transaction details of this smart contract:

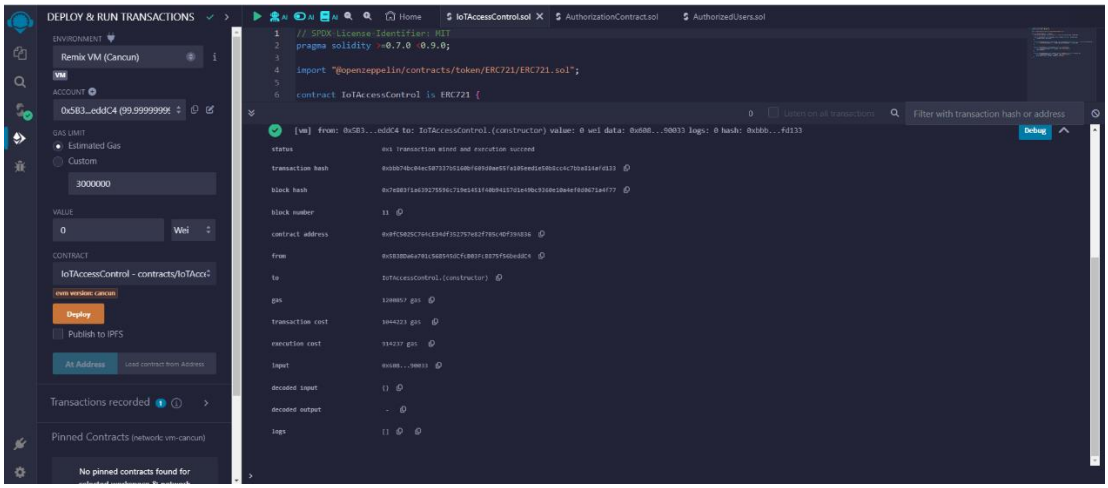

Figure 4.8: The contract transaction (IoTAccessControl)

And here is the result of the contract test (IoTAccessControl):

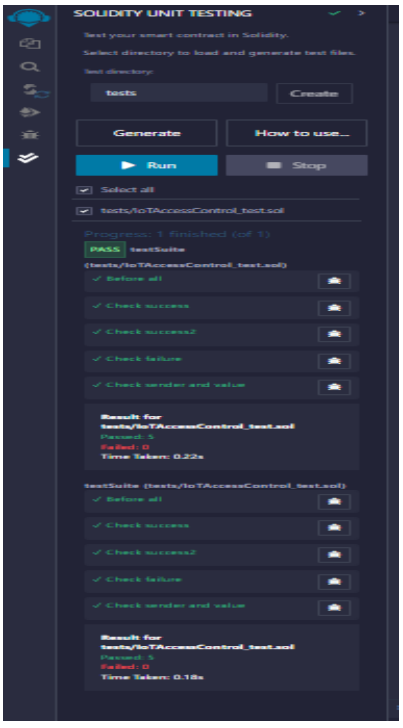

Figure 4.9: Contract test (IoTAccessControl)

Contract execution (IoTCloudInteraction):

You must specify an address see details. This address is the transaction address of the previous contract (IoTAccessControl.sol).and here are the details:

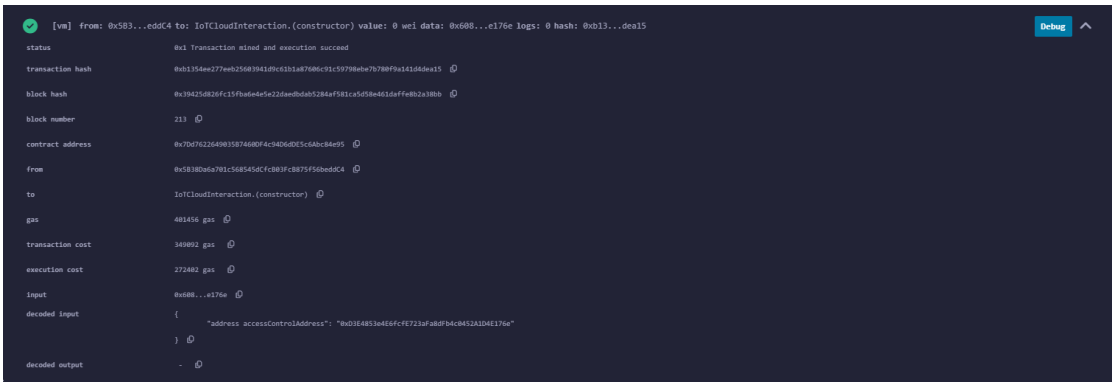

Figure 4.10: The contract transaction (IoTCloudInteraction)

And here is the test of this contract:

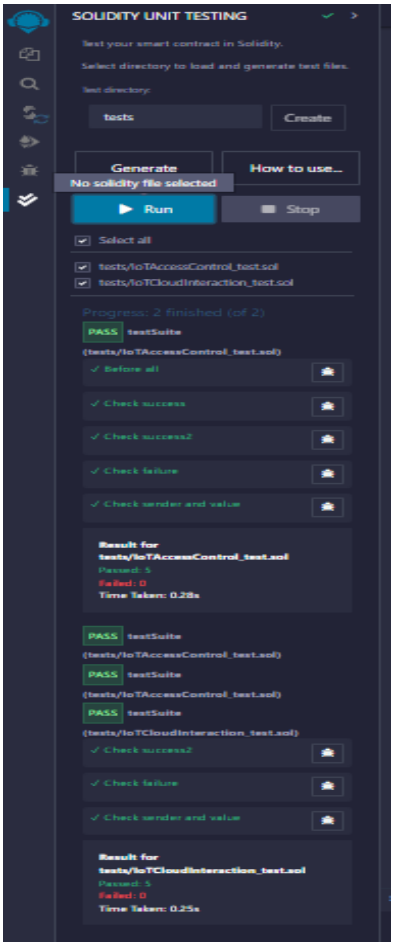

Figure 4.11: Testing the contract (IoTCloudInteraction).

Deployment of the contract (AuthorizationContract):

Here are the transaction details of this smart contract:

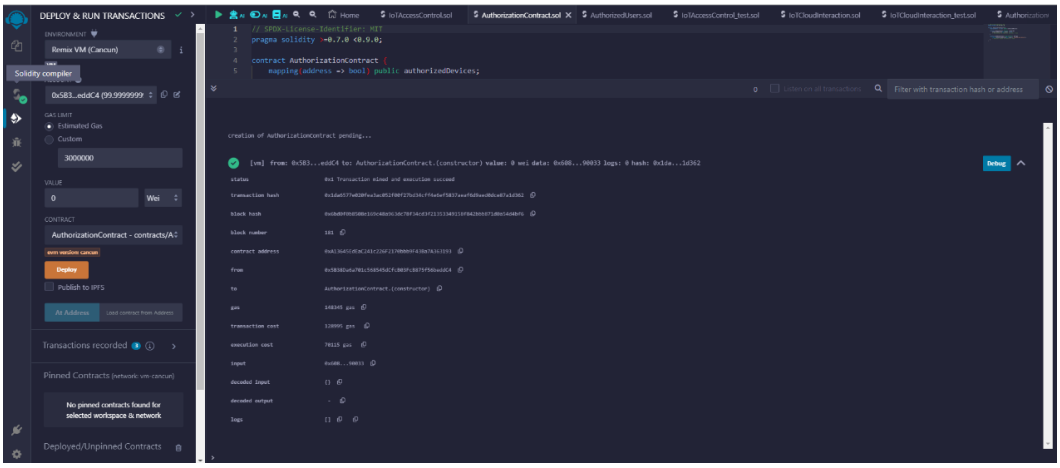

Figure 4.12: The contract transaction (AuthorizationContract)

And here is the test of this contract:

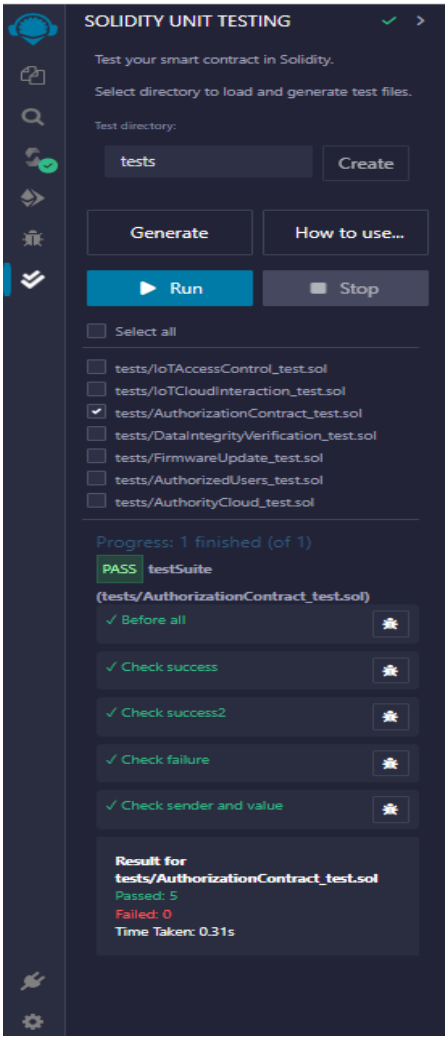

Figure 4.13: Testing the contract (AuthorizationContract).

Deployment of contract (DataIntegrityVerification):

Here are the transaction details of this smart contract:

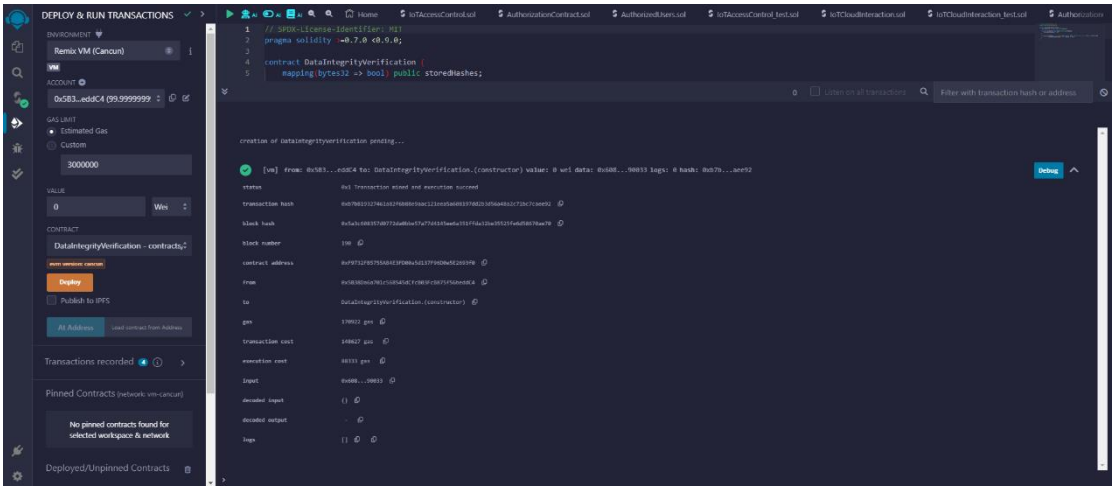

Figure 4.14: The contract transaction (DataIntegrityVerification)

And here is the test of this contract:

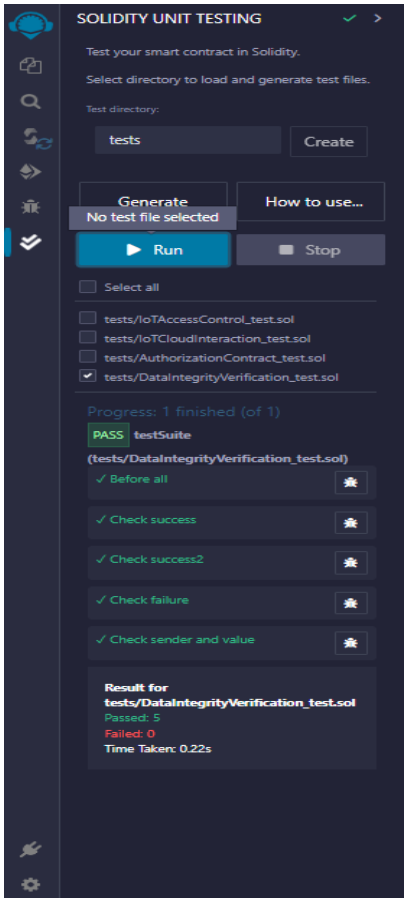

Figure 4.15: Contract test (DataIntegrityVerification).

Deployment of the contract (FirmwareUpdate):

Here are the transaction details of this smart contract:

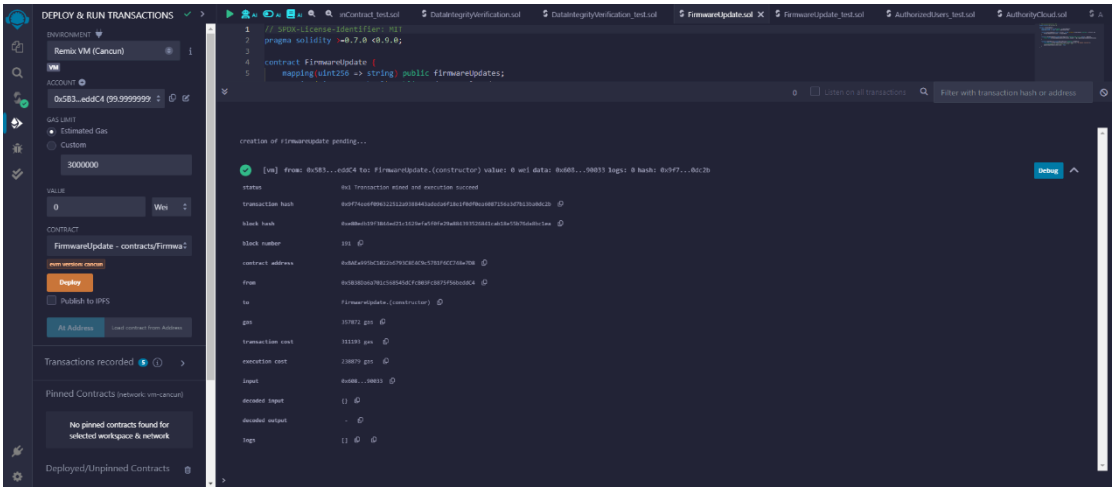

Figure 4.16: The contract transaction (FirmwareUpdate)

And here is the test of this contract:

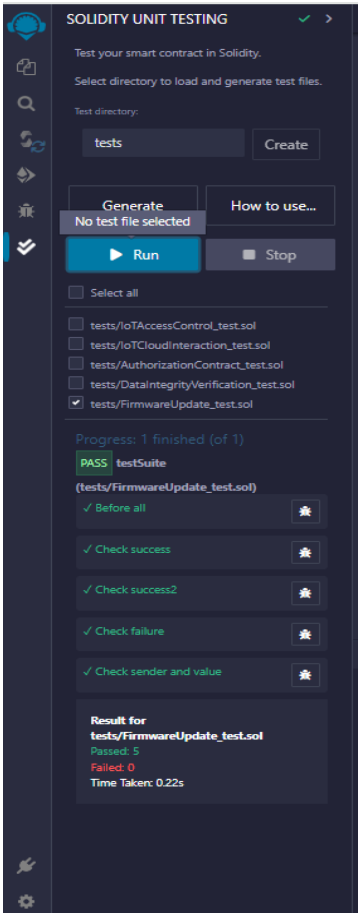

Figure 4.17: Contract test (FirmwareUpdate).

Deployment of contract (AuthorizedUsers):

Here are the transaction details of this smart contract:

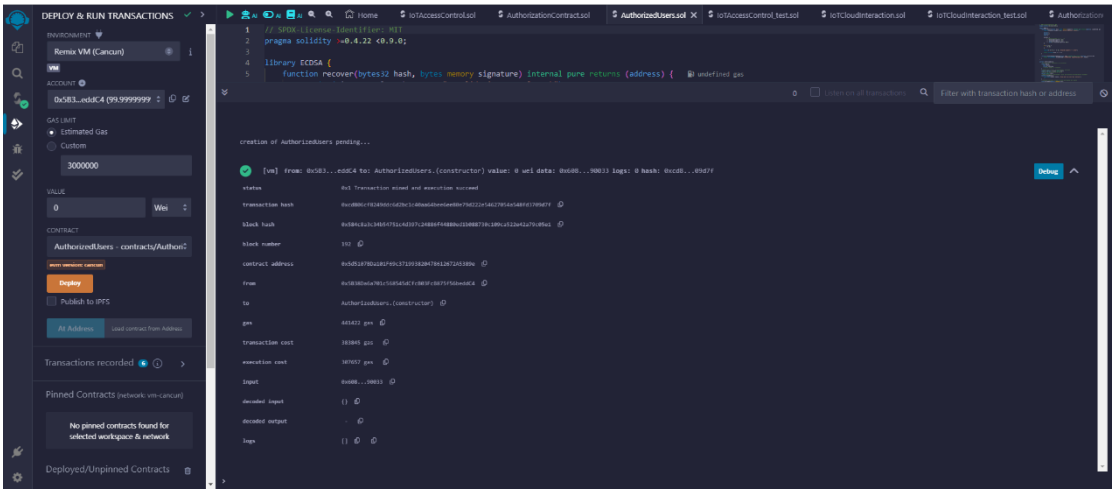

Figure 4.18: The contract transaction (AuthorizedUsers).

And here is the test of this contract:

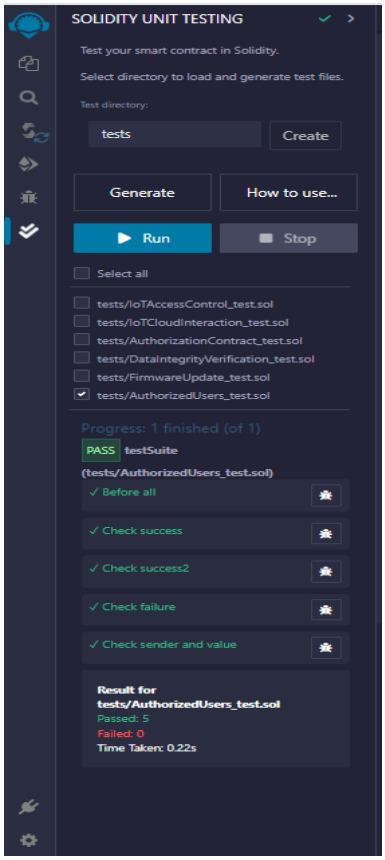

Figure 4.19: Contract test (AuthorizedUsers)

Deployment of the contract (AuthorityCloud):

Here are the transaction details of this smart contract:

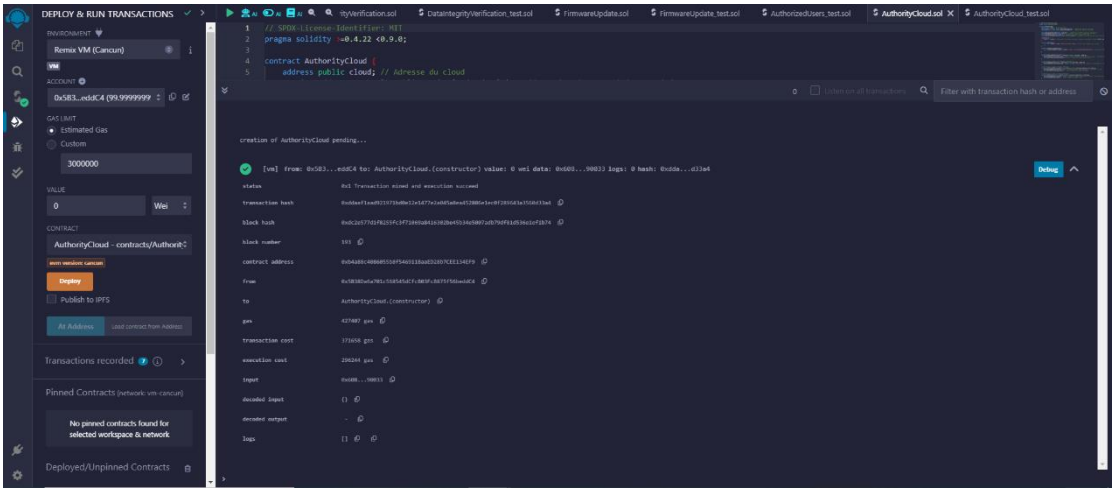

Figure 4.20: The contract transaction (AuthorityCloud).

And here is the test of this contract:

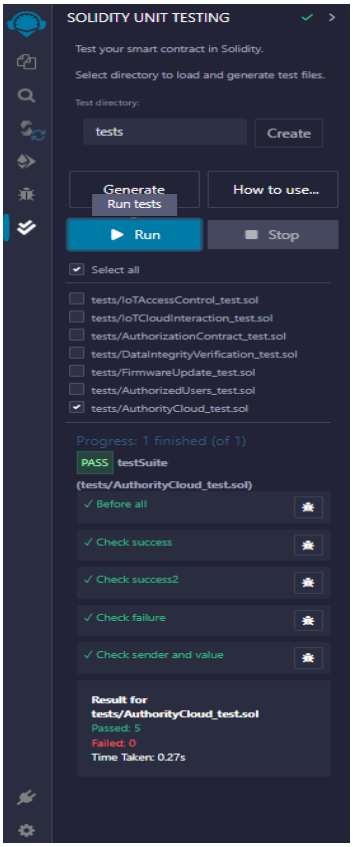

Figure 4.21: Contract test (AuthorityCloud).

5. Performance evaluation:

In this part, we will evaluate the implementation of our system. This evaluation based on gas and cost and speed used in contract deployment and function calls.

Here is the evaluation in detail:

Deployments	**Gas used**	**Total Cost (ETH)**	**Speed**
IoTAccessControl.sol	1044223	0.01119301	0.18s
IoTCloudInteraction.sol	349092	0.01239311	0.25s
AuthorizationContract.sol	128995	0.01438969	0.31s
DataIntegrityVerification.sol	148627	0.01298297	0.22s
FirmwareUpdate.sol	311193	0.01337772	0.22s
AuthorizedUser.sol	383845	0.01525285	0.22s
AuthorityCloud.sol	371658	0.01253068	0.27s

Table 4.1: Evaluation of Deployment of Smart Contracts

Analyzing the performance of the seven smart contracts reveals distinct patterns in resource consumption and deployment efficiency. The total gas used across all contracts amounts to 2,737,633 units, with an overall cost of approximately 0.09212 ETH. On average, each contract requires about 391,090 gas units and costs around 0.01316 ETH to deploy, with a mean deployment speed of 0.2386 seconds. Among the contracts, AuthorizedUser.sol stands out with the highest deployment cost of 0.01525285 ETH, whereas IoTAccessControl.sol is the most economical, costing only 0.01119301 ETH, and it also boasts the fastest deployment time of 0.18 seconds. Conversely, AuthorizationContract.sol is the slowest to deploy, taking 0.31 seconds.

6. Comparison between the proposed solution and the other work:

In this table, we will present a comparison between the results obtained by our solution and those achieved by various users of Blockchain technology in their works.

Performances	**Our solution**	**Work [24]**	**Work [25]**	**Work [26]**
Security	Secured by Blockchain, difficult to manipulate. Access tokens restrict unauthorized users.	Security is high through the use of key management mechanisms.	Security is high as several measures have been taken like robust cryptography and the use of Blockchain.	This work appears to be a significant measures to ensure a high level of security.

Key management	Access tokens can simplify permission management, but blockchain key management remains crucial.	Participating Blockchains (BPs) generate, store, and update keys on the Blockchain.	Key management is a shared responsibility between several entities.	Key management is high, through the use of master keys, random parameters, and transformation keys specific to user attributes.
Response time	The results demonstrated notable speed in the response time of each transaction. Each operation is processed almost instantly, which ensures optimal fluidity and efficiency.	The response time of Blockchain participants depended on the number of Blockchain participants in the system.	The response time depends on the complexity of the image processing algorithms, the speed of data transmission, and the effectiveness of the encryption.	Response time is influenced by encryption and decryption processes attribute matching and interactions with the Blockchain and concurrent request management.
Storage capacity	All data collected by fog computing is stored in the Cloud, while transactions are recorded in the Blockchain.	Data is stored on the Blockchain and participating Blockchains act as nodes dedicated to storing and managing transactions on the Blockchain.	The data is stored locally on the equipment.	Data is stored on a permissioned Blockchain, where all publication and subscription transactions are recorded with timestamps.

Table 4.2: Comparison between the proposed solution and the other work.

Smart contracts with access tokens offer several benefits in terms of security, confidentiality, integrity, and authenticity through their use of Blockchain technology and tokens to control access:

Security: Security enhanced by Blockchain and access tokens limit unauthorized users.

Confidentiality: Access tokens provide high confidentiality by limiting access to authorized entities only.

Integrity: The immutable nature of Blockchain guarantees the integrity of transactions.

Authenticity: Digital signatures on the Blockchain and access tokens verify the identity of the parties involved.

7. Conclusion:

In this chapter, we detailed the implementation and evaluation of our secure communication system for IoT environments based on Blockchain technology and smart contracts. We began by describing the tools and technologies used, including Solidity and Remix Ethereum IDE, for smart contract development. We then presented the implementation of several smart contracts designed to manage the access, storage, and verification of data from IoT devices. Each contract has been tested to ensure its proper functioning and effectiveness in terms of security and performance.

Test results demonstrated that our solution significantly improves the security of IoT communications, ensuring the confidentiality, integrity, and authenticity of the data exchanged. Additionally, the use of Blockchain has added a layer of trust and transparency, making data tamper-proof and traceable. Finally, we compared our solution to other existing approaches, highlighting its advantages in terms of security and data management. This evaluation confirmed that our system offers a robust and innovative approach to securing IoT environments while being adaptable to the constraints of IoT devices.

In conclusion, this chapter demonstrated the feasibility and effectiveness of our proposed architecture, while paving the way for future improvements and research in the area of IoT security.

General conclusion

Conclusion:

The main objective of our work was to present a secure communication system in an IoT environment, based on Blockchain and smart contracts. This system provides enhanced security through the use of Blockchain for data and transaction management, as well as smart contracts to automate agreements and ensure their secure execution.

We sought to provide an innovative and promising solution to address security challenges specific to IoT environments. By combining the advantages of Blockchain and smart contracts, our system offers a robust and reliable approach to ensuring the confidentiality, integrity, and reliability of communications and data exchanges in IoT environments.

In the first chapter, we started by laying out the basics of IoT environments, highlighting the security challenges they face, particularly in terms of data confidentiality, integrity, and availability.

In the second chapter, we examined existing security mechanisms for communication in IoT environments. This literature review allowed us to understand the advantages and limitations of current approaches, and to identify opportunities to develop a more robust solution.

In the third chapter, we presented an overview of Blockchain technology and proposed a security architecture based on Blockchain and smart contracts to improve communication systems in IoT environments. We have described in detail the implementation and evaluation of this architecture, developing numerous smart contracts.

In conclusion, this research paper makes a significant contribution to the exploration of secure communication solutions based on Blockchain and smart contracts in IoT environments. Our proposed security architecture has demonstrated its effectiveness in solving specific security challenges, particularly data confidentiality, and transaction integrity.

Perspective:

In this research paper, we explored and proposed a solution to improve the security of communication systems in IoT environments, using Blockchain and smart contracts.

To improve the security and efficiency of communication systems in IoT environments using Blockchain and smart contracts, several avenues can be explored. It is crucial to develop more lightweight and efficient consensus algorithms for resource-constrained IoT devices, such as Proof of Authority (PoA). Interoperability and standardization of protocols are necessary for smooth communication between different IoT platforms and Blockchain networks. The scalability of Blockchain systems must be improved to handle an increasing number of devices and transactions, using techniques like sharding and layer 2 technologies. The integration of artificial intelligence with Blockchain will enable automation and optimize decision-making processes, by improving data management and security. Continuous research and close collaboration between researchers, industry, and regulators are essential to fully exploit these technologies and create more secure and efficient IoT environments.

Reference :

[1] Gokhale, P., Bhat, O., & Bhat, S. (2018). Introduction to IOT. *International Advanced Research Journal in Science, Engineering, and Technology* , *5* (1), 41-44.

[2] u Farooq, M., Waseem , M., Mazhar , S., Khairi , A., & Kamal, T. (2015). A review of the Internet of Things (IoT). *International Journal of Computer Applications*, *113* (1), 1-7.

[3] https://iotindustriel.com/iot-iiot/architecture-iot-lessentiel-a-savoir/

[4] Sharma, S., & Kumar, S. (2020, January). A review on IoT: Protocols, architecture, technologies, application and research challenges. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)* (pp. 559-564). IEEE.

[5] Bisdikian , C. (2001). An overview of the Bluetooth wireless technology. *IEEE Communications Magazine*, *39* (12), 86-94.

[6] Ramya , C. M., Shanmugaraj , M., & Prabakaran , R. (2011, April). Study on ZigBee technology. In *2011 3rd International Conference on electronics computer technology* (Vol. 6, pp. 297-301). IEEE.

[7] Coskun , V., Ozdenizci , B., & Ok, K. (2013). A survey on near-field communication (NFC) technology. *Wireless personal communications*, *71* , 2259-2294.

[8] Tournier, J., Lesueur, F., Le Mouël , F., Guyon, L., & amp ; Ben- Hassine , H. (2021). A survey of IoT protocols and their security issues through the lens of a generic IoT stack. Internet of Things, 16, 100264.

[9] Chaudhari , B.S., Zennaro , M., & Borkar , S. (2020). LPWAN technologies: Emerging application characteristics, requirements, and design considerations. *Future Internet* , *12* (3), 46.

[10] Yu, YJ, & Wang, J.K. (2018, November). Uplink resource allocation for narrowband Internet of Things (NB-IoT) cellular networks. In 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC) (pp. 466-471). IEEE.

[11] Roberts, C. M. (2006). Radio frequency identification (RFID). *Computers & security* , *25* (1), 18-26.

[12] Stanivuk , I., Bjelić , V., Samardžić , T., & Simić , Đ. (2017, October). Expanding lua interface to support HTTP/HTTPS protocol. In 2017 13th International Conference on Advanced Technologies, Systems and Services in Telecommunications (TELSIKS) (pp. 407-410). IEEE.

[13] Naik , N. (2017, October). Choice of effective messaging protocols for IoT systems: MQTT, CoAP , AMQP and HTTP. In *2017 IEEE international systems engineering symposium (ISSE)* (pp. 1-7). IEEE.

[14] Triantafyllou , A., Sarigiannidis , P., & amp ; Lagkas , T.D. (2018). Network protocols, schemes, and mechanisms for internet of things (iot): Features, open challenges, and trends. Wireless communications and mobile computing, 2018.

[15] Haxhibeqiri , J., De Poorter , E., Moerman , I., & Hoebeke , J. (2018). A survey of LoRaWAN for IoT: From technology to application. *Sensors* , *18* (11), 3995.

[16] Khan, Y., Su'ud , MBM, Alam , MM, Ahmad, SF, Ahmad, AYB, & Khan, N. (2022). Application of Internet of Things (IoT) in sustainable supply chain management. *Sustainability* , *15* (1), 694.

[17] Zhao, W., Wang, C., & Nakahira , Y. (2011, October). Medical application on internet of things. In *IET international conference on communication technology and application (ICCTA 2011)* (pp.

660-665). IET.

[18] Lakhwani , K., Gianey , H.K., Wireko , J.K., & Hiran , K.K. (2020). *Internet of Things (IoT): Principles, paradigms and applications of IoT* . BpbPublications .

[19] Mekala , M.S., & Viswanathan , P. (2017, August). A Survey: Smart agriculture IoT with cloud computing. In *2017 international conference on microelectronic devices, circuits and systems (ICMDCS)* (pp. 1-7). IEEE.

[20] Chaudhary, S., Johari , R., Bhatia, R., Gupta, K., & Bhatnagar , A. (2019, April). CRAIoT : concept, review and application(s) of IoT. In 2019 4th international conference on internet of things: Smart innovation and uses (IoT-SIU) (pp. 1-4). IEEE.

[21] Neshenko , N., Bou-Harb , E., Crichigno , J., Kaddoum , G., & Ghani , N. (2019). Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations. *IEEE Communications Surveys & Tutorials* , *21* (3), 2702-2733.

[22] Deogirikar , J., & Vidhate , A. (2017, February). Security attacks in IoT: A survey. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)* (pp. 32-37). IEEE.

[23] Ding, Z., He, D., Qiao , Q., Li, X., Gao , Y., Chan, S., & Choo , KKR (2023). A lightweight and secure communication protocol for the IoT environment. *IEEE Transactions on Dependable and Secure Computing.*

[24] Kandi , MA, Kouicem , DE, Lakhlef , H., Bouabdallah , A., & Challal , Y. (2020, December). A blockchain -based key management protocol for secure device-to-device communication in the internet of things. In *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)* (pp. 1868-1873). IEEE.

[25] Li, Y. (2023, April). Research on Secure Interactive System of Video Surveillance Data. In *2023 IEEE 12th International Conference on Communication Systems and Network Technologies (CSNT)* (pp. 720-727). IEEE.

[26] Jiang, B., He, Q., Liu, P., Maharjan , S., & Zhang, Y. (2023). Blockchain empowered secure video sharing with access control for vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems* .

[27] Huang, X., Arnold, D., Fang, T., & Saniie , J. (2021, September). A Novel Encryption/Decryption Framework for Ultrasonic Secure Video Transmission. In *2021 IEEE International Ultrasonics Symposium (IUS)* (pp. 1-3). IEEE.

[28] Madani , B., Azzaz , MS, Sadoudi , S., & Kaibou , R. (2023, March). FPGA Implementation of Secure Video Network Communication using Chaotic Cryptosystem. In *2023 International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS)* (pp. 1-6). IEEE.

[29] Naser , HA, Ali, HH, & Ali, HR (2023, February). Utilizing a High-Sensitive and Secure Communication System for Data Transmission. In *2023 Second International Conference on Advanced Computer Applications (ACA)* (pp. 1-5). IEEE.

[30] Umadevi , R. (2016, March). Joint approach for secure communication using video steganography: Achieving better communication based on video steganography. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 3104-3106). IEEE.

[31] Sharma, K., Chaurasia , K., Aggarwal, A., & Sahani , R. (2023, August). PixelVerse: Secure and Efficient File Storage and Transfer through Video Embedding. In *2023 International Conference on Electrical,*

Electronics, Communication and Computers (ELEXCOM) (pp. 1-5). IEEE.

[32] Li, J., Li, Y., Zhang, Y., Li, F., Song, X., & Mao, J. (2023, October). Secure Authentication Scheme for Large-scale Video Surveillance System Based on Quantum Key. In *2023 IEEE 14th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 201-205). IEEE.

[33] Elshamy , AM, Abdelghany , MA, Alhamad , AQ, Hamed , HF, Kelash , HM, & Hussein, AI (2017, September). Secure implementation for video streams based on fully and permutation encryption techniques. In *2017 International Conference on Computer and Applications (ICCA)* (pp. 50-55). IEEE.

[34] Sarma , HKD (2018, December). Secure Multimedia Communication over Mobile Ad-hoc Networks. In *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 812-817). IEEE.

[35] Pai , TP, Raghu, ME, & Ravishankar , KC (2014, December). Video Encryption for Secure Multimedia Transmission-A Layered Approach. In *2014 3rd International Conference on Eco-friendly Computing and Communication Systems* (pp. 127-132). IEEE.

[36] Anoop , BN, George, SN, & Deepthi , PP (2014, July). Secure video transcoders based on correlation preserving sorting algorithm. In *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)* (pp. 771-775). IEEE.

[37] Go, K., Lee, I.G., Kang, S., & Kim, M. (2020). Secure video transmission framework for battery-powered video devices. *IEEE Transactions on Dependable and Secure Computing* , *19* (1), 275-287.

[38] Solidity — Solidity 0.6.8 Documentation (solidity-fr.readthedocs.io).

[39] Remix - Ethereum IDE & community (remix-project.org). gv .