Order No :
Serial No :

People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of El Oued

Faculty of Exact Sciences

Department of Computer Science

Doctorate science thesis in Computer Science

The Security and efficient management of distributed behavior in the context of Internet of Things (IoT)

By
Saci Medileh

Presented on: 22/12/ 2022

Dr Laouid Abdelkader, University of El Oued, Director
Pr. Bounceur Ahcène, University of Western Brittany, France Co- Director

Academic Year: 2022/2023

People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of El Oued
Faculty of Exact Sciences
Department of Computer Science

Doctorate science thesis in Computer Science

The Security and efficient management of distributed behavior in the context of Internet of Things (IoT).

By
Saci Medileh

Dr. Abdelkader Laouid (director)
Dr. Ahcene Bounceur (co-director)

Academic Year: 2022/2023

I would dedicate my dissertation to my wonderful parents in appreciation for their unending love and support throughout my life. To my beloved wife and wonderful children to express my gratitude and admiration for them. Great thanks to all my dear friends and colleagues for their encouragement and advice.

Acknowledgements

First, praise and thank God, the Almighty, who has granted me countless blessings, knowledge, and opportunities throughout my research to finally accomplish the thesis.

I want to acknowledge and extend my sincere gratitude to my research director, Dr. Abdelkader Laouid, who made this work possible with patience, motivation, enthusiasm, and experience. His guidance and advice carried me through all the stages of the research and writing of this thesis. I would also like to thank my co-director Dr. Ahcène Bounceur for his remarkable cooperation and support even though he is abroad. I would like to extend my sincere thanks to my committee members, despite their busyness, for agreeing to participate in the review and improvement of this work.

Finally, to my collaboration teamwork, Mostefa Kara, Mohammad Hammoudeh, Reinhardt Euler, El Moatez Billah Nagoudi,Mohammed Amine Yagoub, Muath Alshaikh, Amna Eleyan. I would like to thank you for your excellent cooperation in our work and your constructive propositions, but mainly for your constant commitment to doing a great job.

Thank You

Saci Medileh

Contents

List of Figures

List of Tables

Abstract

The Security and efficient management of distributed behavior in the context of Internet of Things (IoT).

When the use of smart devices and networks becomes common, the Internet of Things (IoT) field allows the direct connection of physical objects to the Internet using microcontrollers. The security and distributed aspects are considered important factors to protect and manage connected objects from malicious people and/or software. Furthermore, the connected objects can be installed in a hostile environment, where they are highly dynamic, prone to faults, and typically deployed in remote and harsh conditions, which leads to implementing the management of distributed behaviors to maintain the performance of the network and to provide an efficient autonomous system.

Therefore, IoT generates important questions and introduces new challenges for security. In fact, security is a major concern when networks are deployed on a large scale. Therefore, looking for adequate security solutions is necessary to ensure that smart devices capture and share information with appropriate security measures.

This work aims to propose and evaluate the standardized, secure schemes for key exchange protocols in modern Internet of Things architectures consisting of objects (sensor nodes), cloud nodes, smartphones, and end-users. Moreover, to implement/integrate of asymmetric security solutions based on Internet Key Exchange (IKE), they will be evaluated in simulation or a scenario on real networking devices.

Keywords: Security; Cryptography; Distributed System; Management of distributed behaviors; Internet of Things; Connected Devices; Smart Devices.

ملخص

الأمان والإدارة الفعالة للسلوك الموزع في سياق إنترنت الأشياء

الاستخدام الشائع للأجهزة الذكية والشبكات، أدى لتوسع وانتشار بيئة أنترنت الأشياء، وأصبح من الممكن التوصيل المباشرة للأشياء المادية بالإنترنت باستخدام المتحكمات الدقيقة. تعتبر الجوانب الأمنية والموزعة عوامل مهمة لحماية وإدارة الكائنات المتصلة من الأشخاص أو البرامج الضارة. علاوة على ذلك، يمكن تثبيت الكائنات المتصلة في بيئة غير موثوقة، حيث تكون هذه الكائنات ديناميكية للغاية، وعرضة للأعطال، وعادة ما يتم نشرها في ظروف بعيدة وقاسية، مما يقود إلى تنفيذ إدارة السلوكيات الموزعة للحفاظ على أداء الشبكة ولتوفير نظام مستقل فعال.

لذلك، تطرح بيئة إنترنت الأشياء أسئلة مهمة وتقدم تحديات جديدة للأمن، في الواقع، يمثل الأمن مصدر قلق كبير عندما يتم نشر الشبكات على نطاق واسع. لذلك، فإن البحث عن حلول أمنية مناسبة أمر ضروري لضمان جمع وتخزين المعلومات من طرف الأجهزة الذكية ومشاركتها مع الاحتياطات الأمنية اللازمة لضمان السرية والموثوقية.

يهدف هذا العمل إلى اقتراح تقنيات وتقييم المخططات الموحدة والآمنة لبروتوكولات تبادل المفاتيح في بنى شبكات إنترنت الأشياء الحديثة التي تتكون من كائنات (عقد استشعار)، وعقد سحابية، وهواتف ذكية، ومستخدمين نهائيين. علاوة على ذلك، لتنفيذ أو دمج الحلول الأمنية غير المتماثلة لتشفير البيانات القائمة على تبادل مفاتيح الإنترنت، وسيتم تقييمها في محاكاة أو سيناريو على أجهزة الشبكات الحقيقية.

الكلمات المفتاحية: الأمان؛ التشفير؛ الأنظمة الموزعة؛ إدارة السلوكيات الموزعة؛ انترنت الأشياء؛ الأجهزة المتصلة؛ الأجهزة الذكية.

Résumé

La sécurité et la gestion efficace des comportements distribués dans le contexte de l'Internet des objets (IoT)

L'utilisation courante d'appareils et de réseaux intelligents a conduit à l'expansion et à la propagation de l'environnement de l'Internet des objets, et il est devenu possible de connecter directement des objets physiques à Internet à l'aide de microcontrôleurs. Les aspects distribués et de sécurité sont des facteurs importants dans la protection et la gestion des objets connectés contre les personnes ou les logiciels malveillants. De plus, les objets connectés peuvent être installés dans un environnement non sécurisé, car ces objets sont très dynamiques, sujets aux pannes, et sont généralement déployés dans des conditions distantes et extrêmes, ce qui conduit à la mise en place d'une gestion distribuée des comportements pour maintenir les performances du réseau et fournir une système autonome efficace.

Par conséquent, l'environnement IoT soulève des questions importantes et présente de nouveaux défis de sécurité. En effet, la sécurité est une préoccupation majeure lorsque les réseaux sont déployés à grande échelle. Par conséquent, la recherche de solutions de sécurité appropriées est essentielle pour garantir que les informations sont collectées et stockées par des appareils intelligents et partagées avec les précautions de sécurité nécessaires pour garantir la confidentialité et la fiabilité.

Ce travail vise à proposer des techniques et à évaluer des schémas standardisés et sécurisés pour les protocoles d'échange de clés dans les architectures de réseau IoT modernes composées d'objets (nœuds de capteurs), de nœuds de cloud, de smartphones et d'utilisateurs finaux. De plus, pour mettre en œuvre ou intégrer des solutions de sécurité de chiffrement asymétrique des données basées sur l'échange de clés Internet, il sera évalué dans une simulation ou un scénario sur des dispositifs de réseau réels.

Mots-clés : Sécurité; Cryptographie ; Système distribué; Gestion des comportements distribués ; Internet des objets ; Appareils connectés; Appareils intelligents.

General introduction

The remarkable development in the information and communication technology field and the widespread use of diverse smart devices through the infrastructure of physical and wireless sensor networks have led to the creation of massive digital data, communications, and exchanges. Furthermore, in addition to the lack of correct management and monitoring of sensor devices, intentional interference when an attacker illegally intercepts the communication systems between these smart devices may also affect users' privacy and other sensitive issues related to the security and protection of users' personal information.

The Internet of Things can be defined in many different ways, and there have already been many definitions that ultimately lead to the same meaning from different perspectives (1; 2). The Internet of Things is defined as the widespread of a large number of things or objects around us, such as; A group of cellular devices, sensors, electronic devices, etc., which can be communicated with each other through the modern wireless network to achieve common goals in several different fields such as; e-health, smart homes and more. The Internet of Things can also be defined as a paradigm in which objects equipped with sensors, actuators, and processors communicate with each other to serve a meaningful purpose. Information security is one of the most critical issues that must be considered during the essential phases of planning and building the Internet of Things to avoid any risk or expected risk related to security and protection. Currently, we note that the Internet of things is present in all aspects and areas of life, meaning that its uses must pass through the Internet's infrastructure via computer networks. This environment constitutes a suitable environment for intruders on the computer network to try to steal information.

Problem statement

Increasing security concerns, such as software vulnerabilities and cyber attacks, can cause many users to refrain from using IoT devices. These IoT security issues are significant for organizations in healthcare, finance, manufacturing, logistics, retail, and other industries that are already beginning to adopt IoT systems.

This work explains IoT security, why it is crucial, and the main challenges to IoT security and attack vectors. We also discuss ways to secure devices, data, and networks within IoT environments. How can we ensure that smart devices capture and share information using appropriate security measures? Moreover, how to propose, evaluate, or adapt standardized and secure schemes, such as asymmetric security solutions based on Internet Key Exchanging protocols, in modern IoT architectures consisting of objects (sensor nodes), cloud nodes, smartphones, and end users.

Structure of thesis

This thesis was divided into four chapters to address the issues raised. The first chapter presents the background of this work and the literature review of distributed systems that help to build the Internet of Things networks. The second chapter will discuss the Internet of Things architecture in detail and the issues of ensuring information and privacy in the IoT network environment, as well as present the various techniques in cryptosystems and how to adopt these techniques in the IoT environment, considering the limitations of the device.

The third chapter is dedicated to the proposed solution to present a new scalable encryption technique based on the key exchange protocol. It will address the steps of developing a new encryption technique that depends on adapting the asymmetric encryption system in the Internet of Things environment, taking into account the constrained devices in the Internet of Things network.

The fourth chapter is a collaborative work with colleagues, where we will propose the use of a homomorphic technique to ensure data confidentiality, for the cloud computing environment and adapting it to being used in the cloud-based Internet of Things environment. Homomorphic Encryption can be used to address security concerns by enabling any third party to apply computational functions on encrypted data without decrypting them beforehand.

In the last chapter, we will conclude this thesis by summarizing all the work mentioned in the thesis and the effectiveness of the results achieved to secure information, devices, and communication in the Internet of Things environment and cloud IoT. We will also present the prospects of the proposed solutions.

Chapter 1

Distributed Computing Environment - an overview

The rapid development in information and communications technologies and the increased deployment of various smart devices through the physical distributed computing infrastructure and sensory networks have led to the increased growth of massive digital data, communications, and exchanges. The process of exchanging data between these smart devices over distributed computing environment may affect users' privacy and other sensitive issues related to the security and protection of users and their personal information, in addition to the lack of correct management and monitoring of sensors devices furthermore intentional interference when an attacker illegally intercept the communication systems between these smart devices.

1.1 What is a Distributed System?

Distributed computing is a group of computers working together at the back-end and appears as a single coherent system to the end user. The computing elements interact together simultaneously, but if one or some fails, the entire system can still work.

Many definitions of distributed systems have been presented in the literature. A distributed system is a collection of computers connected over a network to transmit messages and communications, coordinate their behavior, and interact to achieve a common goal. A distributed system can be an arrangement of different configurations, such as mainframes, computers, workstations, and minicomputers.(3; 4). According to the above definition, distributed systems have two distinctive features. The first is that each computing station called a "node" can take action and operate independently of the other (Figure 1.1). Generally, the node can be a computer, physical server, workstation, software processes, etc.

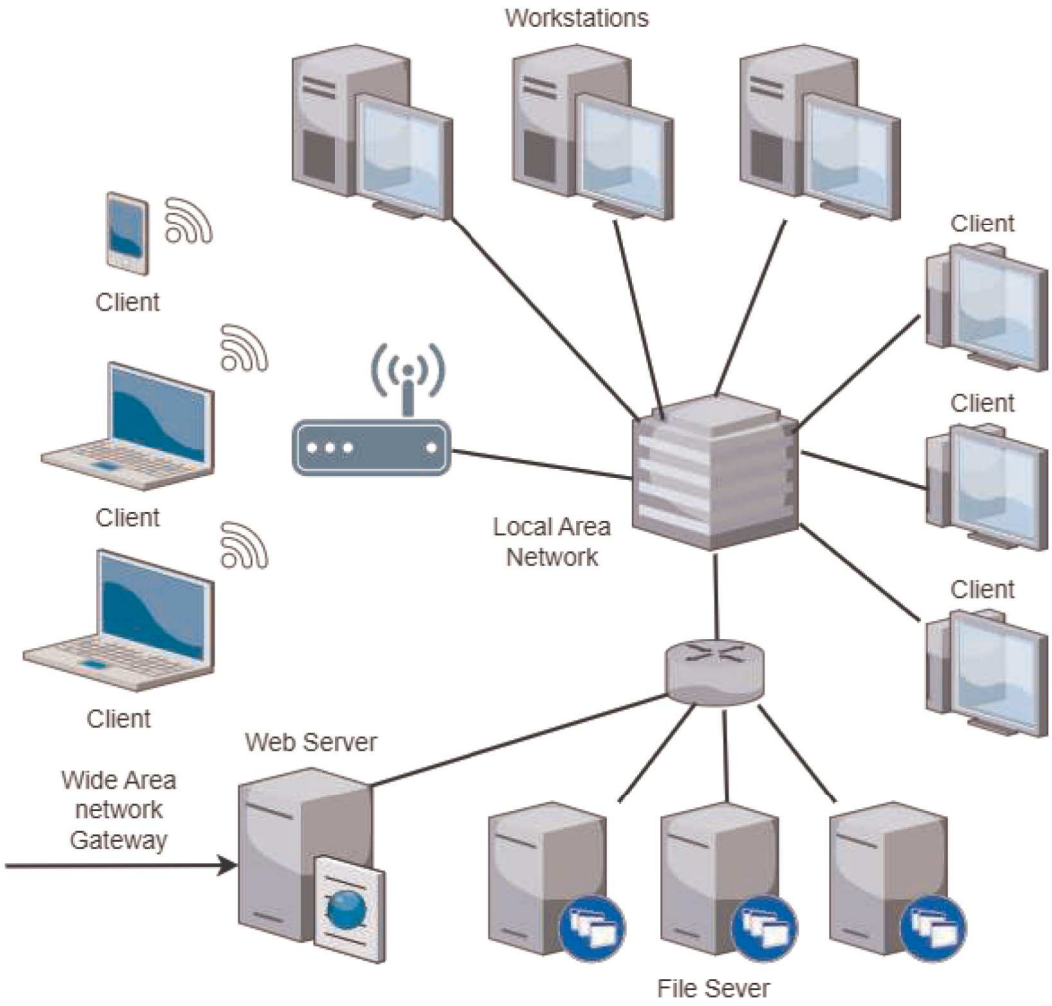


Figure 1.1: A distributed system architecture

The second element is that users think they are dealing with a single system. This means that the autonomous nodes must collaborate in some manner without taking into account any presumptions regarding the nature of the nodes or their connectivity. The size of a distributed system (the number of nodes) can differ from a handful of elements to millions of elements.

The interconnection way in the network can be wireless, wired, or a combination of these two types. Additionally, a distributed system is often very dynamic, if elements can join and leave at any time, with the network topology and underlying performance changing almost continuously. Examples of distributed systems: Networks (local area networks), Telecommunication networks (Telephone networks, VOIP, etc.), Distributed Real-time Systems (such as manufacturing

plants use automation control systems, airlines use flight control systems, logistics and e-commerce companies use real-time tracking systems, etc.), Parallel Processing (such as processors, cloud services, etc.), Distributed Database Systems (that is located over multiple servers and/or physical locations), Distributed artificial intelligence (parallel processing to learn and process extensive data sets using multi-agents).

1.2 Component behavior in distributed systems

In a distributed system, nodes (computers or devices) are independent, which means they can operate independently. Nodes are programmed to achieve common goals by exchanging messages with each other. So when a node receives a message, it processes it and then sends it. Due to dealing with independent nodes, each node in a distributed system has its independent clock. Because there is no one clock that all the nodes must synchronize their behaviors, there are issues with coordinating events and synchronization within a distributed system (3). On another side, autonomous behaviors can ensure the system's security. An autonomous system can make its own decisions or take its actions without human supervision or control (5).

In an autonomous distributed system (6), each node has at least six capabilities required in the decision-making and action process sequence:

- Recognition of the the environment and whole or partial state,
- A self-interested intent proposal,
- Evaluation of all expected result of the intention taking into account suggestions made by other nodes,
- Planed action based on the final decision,
- Learning to enhance decision-making and coordination starigies,
- Synchronization ith other nodes.

In cloud computing, there is another concept of the meaning of autonomy. The intent is to perform operations on data without the need for human intervention. That is, without the need to use it to decrypt it by its owner.

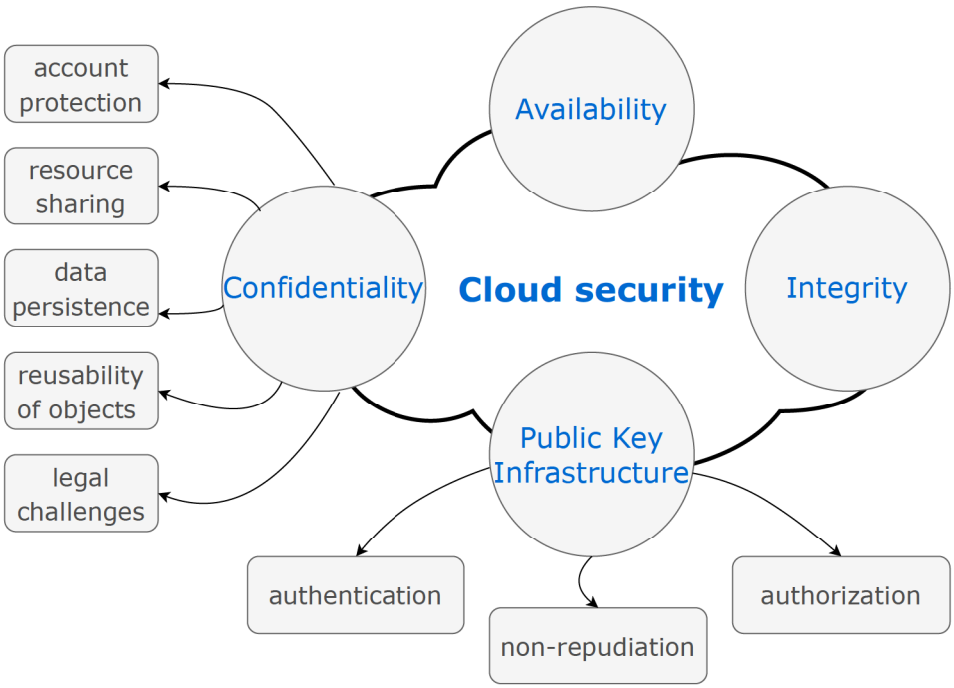


Figure 1.2: Security components in Distributed cloud environment

1.3 Security concerns in Distributed Systems

Since integrating various components in a distributed manner presents new security problems and issues, security is one of the main concerns in developing trusted distributed systems today. It is best to investigate security issues and solutions in these technologies in greater detail to provide a perspective on distributed system security issues (7). To reach the highest level of security in distributed systems, the confidentiality, availability, and integrity of any data shared, transmitted, or stored in the separate systems must be ensured. Access control to data and services or any components should be restricted. Data and processes must be separated at the level of cloud systems to prevent data leakage between applications or users (8; 9). The security, in general, is related to the three major categories of assets that must be protected, data, software, and hardware resources. (Figure 1.2).

1. Confidentiality

    - Resources are shared: Several aspects are shared, memory, programs, etc. So we are talking about multitasking, which can present a certain number of threats.

- Reusability of objects: Reusable objects can create a severe vulnerability in data extraction.
- Data remanence: The remanence of data can lead to inadvertent disclosure of personal data; an attacker can reclaim a large amount of disk space and then recover sensitive data.
- Account protection against theft (usurpation): Without strong authentication, an unauthorized access to users' accounts can be done.
- The confidentiality of software that processes personal data: Applications offered by the organization that owns the infrastructure must be certified.
- Legal challenges: Data storage in multiple locations creates privacy issues.

2. Integrity:
   Integrity means that data, software, and hardware can only be modified by authorized parties and in an authorized manner.

3. Availability:
   To achieve availability, the system must continue to work even in the event of a security breaking.

A distributed environment can be used by a trusted third party to solve specific security issues, enabling secure transactions between many users. This third party relies on PKI (Public Key Infrastructure) (10; 11; 12). Public Key Infrastructure can be a good fit for enabling secure and trusted communication between different components within a distributed environment if planned correctly and practices this list of concepts: (13; 14):

- Authentication is the act of proving an assertion, such as the identity of a computer system user. Authentication simply means the identification of parties involved in the transaction.
- Authorization is the function of specifying that an authenticated user has access rights/privileges to resources.
- Accounting is the process of tracking user activity while accessing network resources.
- Confidentiality is the act of preserving restrictions on unauthorized parties accessing information.
- Data integrity refers to the accuracy and consistency of data, preventing unauthorized information modification.

- Non-repudiation is a technique that uses a digital signature or encryption to ensure message transmission between parties. Any situation in which a person cannot deny the authorship of a document or its existence in a location..

1.4 The Internet of Things (IoT) Paradigm

The internet of things (IoT) was first coined in 1999 by a British visionary called Kevin Ashton. Since then, technology and market trends related to IoT have skyrocketed, and by 2025 according to a report from McKinsey Global Institute, Internet of things applications could have $11 trillion impact (15; 16). The current Internet architecture that has revolutionized the way information is exchanged is about five decades old since it was conceived. There have been many important advancements be it the World Wide Web (WWW), the rise and growth of cellular Internet from GPRS to edge to the current LTE (we are talking about 6G at the time of writing this book). However, the massive use of Internet was envisioned when things were given the intelligence to be able to connect to its peers and communicate over a network (17). The Internet of Things (IoT) paradigm as it is called today opened up a plethora of applications and opportunities for the devices. IoT also brought about the first need for context-aware and content-aware computing. The computing world now considered the interconnectivity between highly heterogeneous devices that exchange real-time information with the need for real-time (on the fly) processing and rendering in some cases. Overall, there is the generation of a large volume of information through the many sensors that are a part of the IoT infrastructure, and it is important for us to create a way to secure it. In IoT, millions of smart devices are connected through the Internet. It is based on the vision of connecting and having access to all real-life physical devices including vehicles, buildings, utility devices, household items, control systems, and other things in real time over the Internet. All the above promises are made based on the rapid advancements made in sensor networks and radio frequency identification (RFID) technology making it an integral part of human life today (18; 19; 20).

1.5 Conclusion

Information security in distributed systems has become the most crucial challenge for distributed systems developers. In this chapter, we discussed the concept of distributed systems and their relation to the rapid development of information and

communication technologies. We presented security concerns in Distributed Systems that arose from sharing and distributing resources between the components of these systems and how their autonomous behaviors can ensure the system's security. We also presented the Internet of Things paradigm and its connection to the development of distributed systems. Thus the security challenges posed by the widespread use of smart devices and their extensive uses in our daily lives and the large amount of information exchanged between these smart objects impose the need to think about novel encryption schemes or to adjust previous cryptography techniques to ensure the privacy and the confidentiality in this environment.

In the next chapter, we will discuss in detail the concept of the Internet of Things, the architecture of IoT networks, and the structure of protocols used for device interconnection in the smart environment. We will present the solutions to securing the IoT environment and the cryptography techniques adopted in this field.

Chapter 2

Securing the Internet of Things for smart environments

2.1 Introduction

The Internet of Things (IoT) is already giving real-world applications, from smart homes and cars to health monitoring and smart utility monitors. All IoT sensors and devices can potentially violate information security rules (confidentiality, integrity, and availability). Using sensors in the IoT environment makes it easier for attackers to track the device user's activities. The widespread use of these devices and allowing them to connect to the internet exposes them to several severe vulnerabilities, putting the user's privacy at significant risk if not appropriately secured. IoT security is very important, as proved by several increased cases in which a common IoT device was used to infiltrate and attack the entire network. IoT security includes a wide range of techniques, strategies, protocols, and actions that aim to mitigate the increasing IoT vulnerabilities of modern businesses.

2.2 Internet of Things security

IoT security refers to the techniques, methods, strategies, and tools used to protect internet-connected devices and networks in the internet of things (IoT) environment from being compromised (17). Using these connected devices by many companies to collect sensitive data and create profiles of people allows them to be hacked by an attacker (20; 21; 22).

In IoT environments, cryptography techniques are the foundation for security and privacy protection because they secure access by unauthorized parties during data processing, transmission, and storage. Due to their high energy consumption and computational complexity, traditional encryption standards and algorithms are

unsuitable for smart devices with limited resources. Therefore, it is necessary to seriously consider using lightweight encryption algorithms. (23)

2.2.1 Securing Smart Devices

Connected IoT devices are rich sources of data collection because they contain one or more sensors and are constantly collecting data. To ensure the security of this collected information, we must consider many possible risks and search for technical solutions to secure the Internet of Things environment and the information obtained through sensors. Consider the possibility that the system has vulnerabilities. In that situation, attackers can easily access connected devices and obtain the required information, which significantly risks users' sensitive data privacy. The system needs to be secured to prevent attacks. Access from unauthorized parties is restricted by data access control. It would also be advantageous if improper use of stored data reduced system control (23; 9; 24).

2.2.2 Challenges

Smart IoT devices can gather and generate massive volumes of data, but they are still severely limited in energy, computing, and storage. Some devices have a limited battery life that provides them power and a limited ability to communicate information. These devices' limitations make them unable to generate keys, encrypt and decrypt information. Since all these operations come at a very high cost, leading to large consumption of its resources, ignoring security and focusing more on improving functionality can extend the life of these resources, but it causes significant security issues (23). Due to the current Internet Protocol (IP) inability to solve all concerns has made interoperability issues a significant challenge due to this situation (23). The presence of some IoT devices that cannot connect directly to the Internet and some of them use other wireless communication protocols such as Bluetooth, ZWave, LoRA, etc., depending on their capabilities and the applications for which they are designed, makes widespread adoption of IoT devices difficult. In addition to these issues, usability presents a significant challenge to IoT device security. The user must manually perform security measures such as bootstrapping, changing passwords, registering keys with a controller, requesting certificates that can be used with the device, and many other tasks in order to secure the IoT device in the current solutions. In most cases, the device is installed in the working environment and allowed to communicate with other devices inside and outside the environment, making it vulnerable to any potential attacks. One of these unprotected devices can endanger all devices, thus compromising the security of the entire system (22; 25).

2.3 Internet of Things Protocol Stack

The Internet of Things is the convergence of embedded systems, wireless sensor networks, and control systems and their use in the smart manufacturing environment, healthcare, smart homes, cities, and wearables (26). IoT technologies enable you to build data-driven systems, improve operational processes, and efficiently use resources. IoT technology continues to expand and evolve, with countless service providers, different platforms, and millions of new devices emerging each year, making it important for developers to consider the many decisions they must make to enter the IoT ecosystem. The IoT technology ecosystem comprises the following layers: devices, data, connectivity, and technology users (17). To plan and build an IoT technology project, it is first necessary to define IoT protocols and requirements to ensure how devices can connect and communicate. In the IoT technologies stack , devices communicate directly or through gateways that can be used to connect IoT devices to the cloud. The data collected from IoT devices is processed through the gateway and then sent to the cloud. The use of IoT gateways provides an additional layer of security to protect data traffic in the IoT network, but this can leads to reduce latency and lower transmission volumes (27; 2; 1).

2.3.1 Current Internet Protocols

IoT protocols are used by IoT devices to communicate. The transmission of data over the Internet is controlled by a set of rules known as the Internet Protocol (IP). IoT protocols make sure that data from a device or sensor can be transmitted, read, and processed by other network objects.Many IoT protocols have been adopted and optimized for a variety of applications. The choice of IoT protocol depends on the device's used connectivity and functionality, the system architecture layer, and the data transmitted over the network.Due to the variety of IoT devices available, the use of the right protocol in the right context is critical for developers of IoT networking systems (20; 23).

There are different architectures for the Internet of Things, which have been proposed by different researchers, of these structures there are two basic architectures: one with three layers and the other with five layers as shown in Figure 2.1. The most basic architecture is a three-layer architecture consists of three layers: perception, network, and application (2; 24).

- **The perception layer:** This layer typically consists of various device types, and the data is collected and sent to the next layer for transport and processing requirements.

- **The network layer** is in charge of transporting and processing sensed data based on the needs and application design, and also maintain the connectivity of smart things, network devices and servers.
- **The application layer** is in charge of delivering the specific requirements and services to users, defining various applications aimed to be used in the IoT network..

The three-layer architecture represents the first and basic description of the IoT protocol architecture. Still, with the expansion of the use of IoT networks, the increase in the size of processed data and its transmission between network components, and the emergence of security challenges to securing data, this three-layer architecture has become insufficient for research on IoT protocol, one of the proposed layer architectures is the five-layer architecture Figure 2.1, which includes in addition to the three layers the processing and business layers. The five layers are perception, transport, processing, application, and business layers. The role of perception and application layers is the same as the three-layer architecture. We define the function of the remaining three layers (20; 2; 13).

- **The transport layer** is responsible for transferring the collected data by the sensors devices to the processing layer and from there to the compute nodes. The transport layer can use a variety of media, such as cellular technology like 3G and LTE modules, LoRA, RFID, NFC, and wireless protocols like Wi-Fi and Bluetooth.
- **The processing layer** is responsible for storing, analyzing, and processing huge amounts of data from the transport layer. It employs many technologies, such as databases, cloud computing, and big data processing modules. It can manage and provide a diverse set of services to the lower layers.
- **The business layer** is responsible for managing the whole IoT system, including applications, business plans, monetization structure of the system, payment models, and users' privacy.

2.4 Foundations of Cryptosystems

A cryptosystem is a way of masking confidential information to be unreadable to unauthorized people. The two most common uses for a cryptosystem are to store data securely in a computer file or to transmit it across an insecure channel, like the internet. Even though the document is encrypted, unauthorized people can still gain access to it, but they would not be able to understand what they

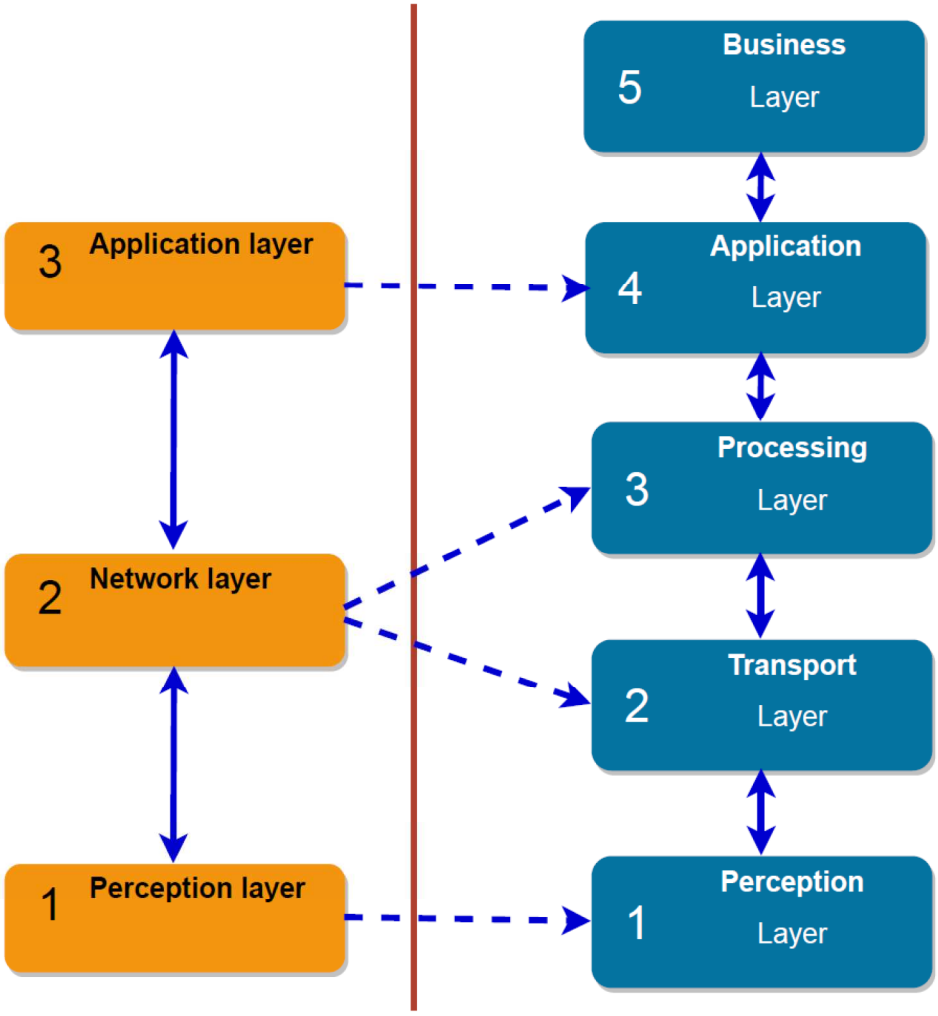


Figure 2.1: The three and five layers architecture of IoT stack

are seeing. The information that needs to be concealed is called plaintext, and the operation of concealing it is known as encryption. The encrypted plaintext is called the ciphertext, and the encryption algorithm is a set of rules used to encrypt the plaintext. The encryption algorithm usually depends on an encryption key, which is input to the algorithm along with the message. For the recipient to be able to read the message, there has to be a decryption algorithm that, when used with the appropriate decryption key, reproduces the plaintext from the ciphertext. The rules that makeup one of these cryptographic algorithms are usually very complicated and must be designed carefully (28; 12). Information confidentiality is ensured through encryption systems, which provide the sender of the information the guarantee that the information is not transmitted in plaintext. Confidentiality, Availability, and Integrity. These three components comprise the information security model's CIA triad. However, there is also a requirement for an operation that can recover the original message if the data has been encrypted using encryption methods.

2.4.1 The field of Cryptography

The cryptography it has evolved over time (29). The earliest known use of cryptography dates back to the times of the Roman emperor Julius Ceasar. Ceaser is expected to have used a simple modification scheme to encrypt his information exchanges with his generals and thus prevent the enemies from identifying the transmitted information. In his honor, we have the famous substitution cipher named "Ceasar Cipher" which is one of the most primitive symmetric key cryptographic schemes (30; 31).

2.4.1.1 Definitions

Cryptography word comes from the Greek word "kruptos" which means to hide and "graphein" which refers to writing. It is literally the art of hiding writing and generally hiding information, either in communication or in storage. Its initial use is in the military field. Armies needed a secure way to exchange sensitive information, such as plans, strategies, etc. After creating computers in the early 1940s and thanks above all to the advent of the Internet in the early 1990s, its use has developed and diversified in almost all areas of public life (32). Today, we find cryptography in everyday objects such as bank cards and smartphones; also during operations performed by computers such as connecting to a secure site, accessing a database, paying for a product on an e-commerce site, etc.

Cryptography is present today to ensure specific objectives such as integrity, confidentiality, authentication, and non-repudiation during all these sortings that we

carry out on data or on the Internet (distributed systems, cloud computing, etc.) almost unconsciously. Cryptography is part of a much larger discipline called cryptology which further contains cryptanalysis. The latter is a discipline that is opposed to cryptography and consists in studying and analyzing the robustness and the weakness of a cryptographic algorithm in order to exploit them and find, for example, the information it protects.

2.4.1.2 Symmetric encryption scheme

The symmetric key cryptography, often known as symmetric encryption, is the use of a single key for both encryption and decryption operations (33; 34; 35; 36). For many years, it was the most popular approach for securing information, and some systems still utilize it with other schemes. The two most known names usually associated with symmetric key cryptosystems are Advanced Encryption Standard (AES) and Data Encryption Standard (DES) (37). For example, Rijndael (AES) (38; 39) presented a multi-turn block cipher with larger and variable block and key sizes. Hill's work (40; 41) uses modulo 26 to encrypt the 26 letters.

A symmetric encryption scheme (42) consists of a pair of algorithms: the encryption algorithm E and the decryption algorithm D. Given a message m and a key k, its cipher is deduced as follows:

$$c = E(m; k) = E_k(m) \tag{2.1}$$

To find the plain text from the ciphertext c, we use the key k and the decryption process D as follows:

$$m = D(c; k) = D_k(c) \tag{2.2}$$

where the only key used is the secret shared between the users. D is necessarily deterministic, that is to say, that to a pair $(c; k)$ it associates a single message m. On the other hand, nothing imposes a priory on E to be so, and this will generally not be the case i.e., it is possible that several values correspond to a pair $(m; k)$ supplied by E, in the case where E is randomized. In this case, it performs an expansion on the size of the ciphertext. The distribution of keys is one of the essential problems of symmetric cryptography, if n people can communicate confidentially, then $n(n-1)/2$ keys are needed. On the other hand, the main disadvantage of symmetric key encryption is that the user must have a secure private canal to transfer the encryption key to the receiver; thus, any key attack will destroy the whole communication. Whereas, with asymmetric encryption, when an adversary discovers the private key of one party only the data exchanged between these two entities will be revealed. Furthermore, symmetric key encryption is weak in digital signatures. Figure 2.2 shows the scenarios and the consequences of using symmetric encryption.

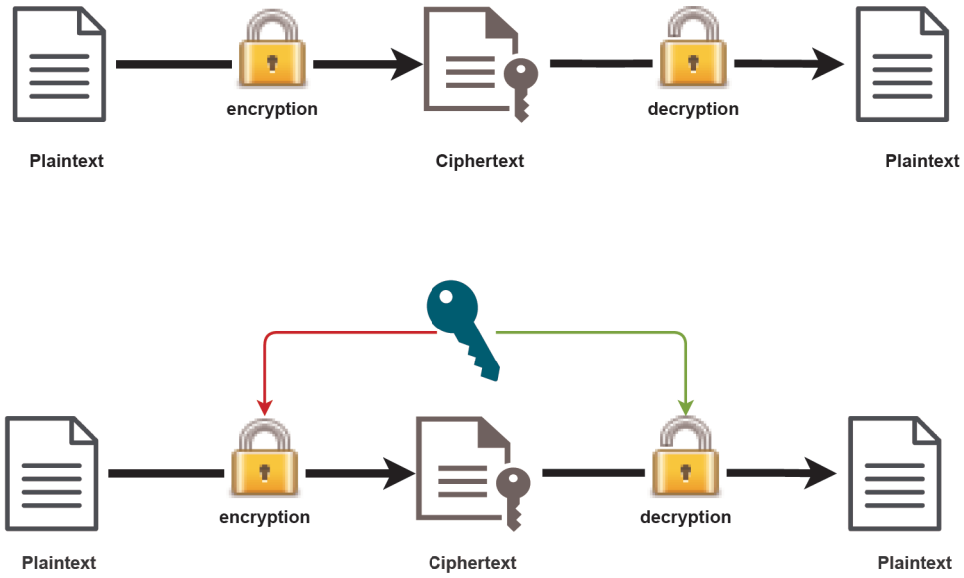


Figure 2.2: Symmetric encryption scheme

2.4.1.3 Asymmetric encryption scheme

The symmetric key cryptosystems have certain weaknesses. To overcome the major disadvantage related to the compromise of one key being a potential threat to the whole system as it becomes a single point of failure, the Asymmetric key cryptographic techniques introduce two sets of keys: one that is publicly available and another that is a secret or private key that is retained by the communicating entities. It is considerably more complex and expensive than symmetric key cryptography. The public key cryptosystem introduced by Diffie-Hellman (31) allows encryption from the recipient's public key. A message that is encrypted using a public key can only be decrypted using a private key, while also, a message encrypted using a private key can be decrypted using a public key. (43; 44; 45). Public-key cryptography is sometimes called "Asymmetric encryption" the term "asymmetric" comes from the difference between private key and public key and includes several important areas such as public-key encryption, signing, exchange of key, etc. (46). The asymmetric technique is based on the one-way functions; applying this function to plaintext is simple but extremely difficult to find it from when it was encrypted (transformed).

In the case of asymmetric encryption, each interlocutor has a pair of keys composed of a public and a private key pk and sk respectively. As with physical keys,

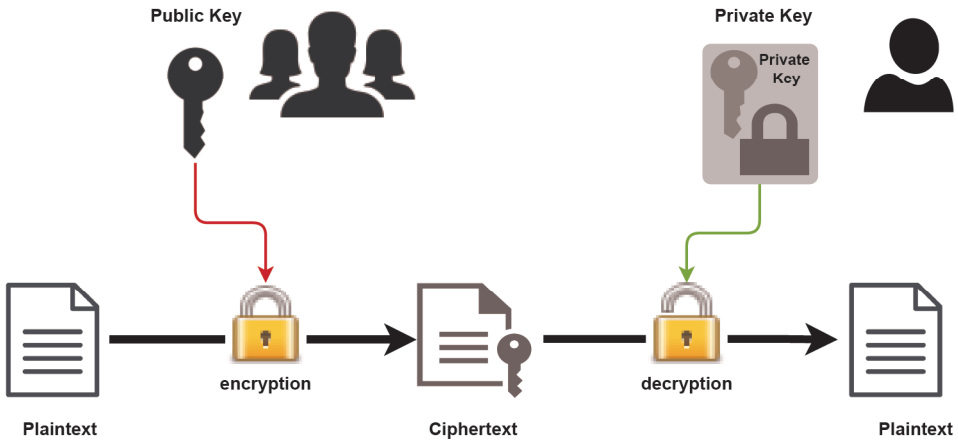


Figure 2.3: Symmetric encryption scheme

pk and *sk* are linked as in a keyring. The two keys are closely linked using a mathematical relation in each cryptosystem. The encrypted data using the public key can only be decrypted using the corresponding private key, Figure 2.3 explains the asymmetric encryption scheme. To ensure data protection and the unimpeded operation of asymmetric encryption, it is necessary to remain *sk* secret and not known to anyone, not even other interlocutors.

The transmission of keys takes place during the first contact. At the same time, the private key creates a digital signature and can thus be identified by other users. In other words, asymmetric encryption allows users to access the public key but can only decrypt messages with the private key. By this, we can ensure a highly secure data exchange. A significant problem of asymmetric schemes is their slowness compared to symmetric cryptography, which makes them to nearly a thousand times faster.

2.4.1.4 Cryptography to secure information

Cryptography is the process of manipulating data using a specialized algorithm, in order to make the data unreadable to anyone who is not authorized to view it. This process is known as ”encryption” and is a key aspect of cryptology.

Encryption schemes enable the sender of a message to be confident that the information cannot be read by anyone who is not authorized to do so, thus ensuring the confidentiality of the information. Confidentiality, integrity, and availability are the three tenets of information security. However, if the information has been transformed into a new form using encryption techniques, there is also a need for a process that can retrieve the original message. The techniques involved in recreating the initial plaintext are studied under another field called cryptanalysis. The specific technique of inferring the message from an encrypted form is called "decryption." The symmetric algorithms are specially used for cipher block encryption. Asymmetric algorithms are generally used for ensuring symmetric key exchange and offering digital signatures (12; 47; 28).

Every use of cryptography (48):

- **Secure transmission:** The main goal of cryptography is to prevent an uninvolved party from reading the data. Most information exchange systems use a secret key scheme. This system uses a common key for encryption and decryption between the interlocutors. Secret keys are exchanged, used, and then destroyed periodically. Each user must secure the secret key from unauthorized access because any third party who has the key can decrypt the data. Therefore, the best way to secure data is to use a public key cryptographic system.

- **Secure storage:** It refers to applying cryptographic schemes to data, both in transit and in storage. Encryption in storage is vastly used by companies that manage storage area networks (SANs). Data secrecy is ensured by storing it in an encrypted form. At the start of a session, the user only has to offer the key to the server for accessing the data and then performs encryption and decryption throughout the use.

- **Integrity in transmission:** Cryptography is also used to offer a way of ensuring that data is not modified during transmission, i.e., data integrity is preserved. For example, in an electronic funds transaction, it is very sensitive that integrity is controlled. A bank can lose billions if an attacker intercepts information. Cryptographic strategies are used to prevent the accidental or intentional change of data during its transmission, which may lead to faulty actions.

- **Integrity in storage:** Access control mechanisms had ensured storage integrity with keys, locks, and other safeguards to avoid unauthorized and suspicious access to stored data. The spread of computer viruses adjusted the scenarios, and there was an urgent need for integrity against intentional attacks.

- **Authentication of identity:** Authentication verifies whether the user has sufficient authority to manipulate or access data. Effective passwords are used to identify someone. Cryptography works the same as the practice of delivering passwords for identity authentication. Passwords are similar to the cryptosystem key that encrypt and decrypt anything the password has access to. The main factor here is the password selection method.

- **Credentialing systems:** Are proof of competence and capability attached to an entity to demonstrate suitability for something. Advancement in the field of implementing electronic credentials has been relatively slow, as the bank checks credentials before agreeing on the loan. Electronic credentials permit electronic validation of the credence of a claim.

- **Digital signatures:** It is a means by which a sent message can be authenticated, i.e., proving that a message comes from a known sender, largely like a signature on a paper document. To make the digital signature more effective than a paper signature, it must be difficult to forge and accepted by a court as binding on all transaction parties. The need for digital signatures appears when the participating users in an exchange are not physically close; additionally, if the volume of paperwork is high, i.e., deals between large companies. Digital signatures can be made by using a hash function and a public-key cryptosystem. The hash delivers a message digest which is a unique small illustration of the original message, where this function is a one-way algorithm; so the message cannot be derived from the digest.

- **Electronic money:** It has replaced cash in financial transactions between users for a long time. The system may use cryptography to maintain individuals' assets in electronic structure. Digital gold currency, electronic funds transfer (EFT), virtual currency, and direct deposit are real models of e-money. EFT electronically exchanges money between two user accounts through computerized systems. This includes ATM withdrawals, online payments, debit card payments, direct deposits, wire transfers, etc. Another application of e-money can be found in e-commerce, and companies such as PayPal mediate the transfer. When a user withdraws money from the bank, the main property of cash is anonymity.

- **Threshold cryptosystem:** Are developed to authorize use only if a number of users above a threshold approve said use. In practice, to decrypt a message, a minimum number of users are required to collaborate between them. If there is less than, the decryption can not be achieved. Such mechanisms spare users from acting alone and at the same time allow other users to be absent without interrupting the operation. Most threshold cryptosystems have keys that are divided into parts. The most ordinary method of splitting a key is to form the key as the solution of an equation of N variables.

- **Secure multi-party computation:** It consists of a set of parties with secret inputs who wish to together compute a function of their data so that keeping certain security properties (correctness, confidentiality, etc.). This kind of computation provides solutions to many practical issues including distributed voting, private auctions, shared signing, decryption functions, and problems requiring private data retrieval. A famous example of secure multiparty computing solves Yao's millionaire problem when two millionaires desire to know which of them is richer but without either showing their net worth to the other.

2.4.1.5 Signature scheme

Let's say user A sends a message to user B. At first, A hashes the message to produce a digest D; $H(m) = D$, then encrypts D using the secret key to create a personal signature S; $Enc(D) = S$. The message M is then encrypted using the secret key K; $Enc(M) = C$. Upon receipt, B decrypts C and S using K; $Dec(C) = m$, $Dec(S) = d$. He would then hash the obtained message m using the same hashing algorithm (whose name was appended in the sent text) with which M had previously been hashed. If the obtained digest $H(m)$ and the decrypted digest d are the same ($H(m) = d$), then the signature has been proved, and B would be assured of the integrity of the message.

In asymmetric techniques, user A encrypts the message M using the public key of user B, but the digest D should be encrypted using his private key. Upon receipt, B decrypts C using his private key and decrypts S using the public key of the sender A. Another characteristic of this mechanism is the non-repudiation of digital signatures. Due to the private key being only accessible to the sender (user A), he cannot deny having signed the message. Further, a digital signature can be verified by anyone who has the sender's public key in the case of asymmetric encryption.

2.4.2 A taxonomy of lightweight cryptography techniques

This section aims to present an overview of lightweight cryptography techniques. The main characteristics of lightweight cryptography are low-cost, performance and applicable security level. An optimal lightweight cryptography has to achieve a good balance between security level, computation time and power consumption. Regarding these characteristics, we can classify the previous work in lightweight cryptography into three main classes: physical primitives, computational primitives and ultra lightweight protocols as shown in Figure 2.4.

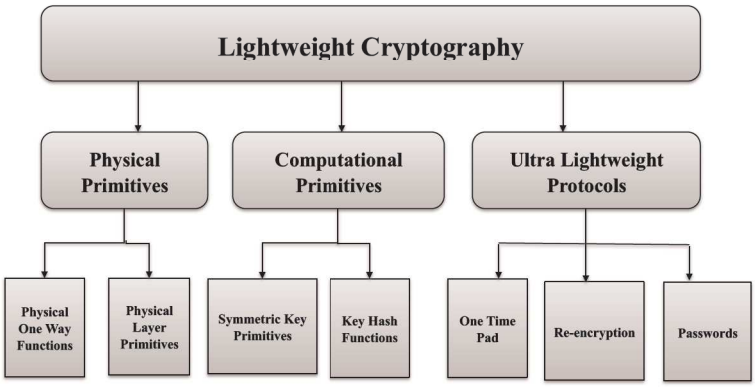


Figure 2.4: A Lightweight Cryptography Taxonomy.

2.4.2.1 Physical primitives

The main principle of physical primitives is the use of bio-metrics for authentication. A physical primitive can extend an analogue singularity or variation in physical features, which is essential to physical structures but excessively hard to duplicate, and translate it into a digital value for the determination of a specific quantification. Physical layer protocols are designed with the aim to minimise power consumption without compromising the fidelity. Power consumption should be linearly scaled as a function of data rate (49). In (50), the authors describe Physical One-Way Functions (POWF) which focus on measuring the accuracy of sprinkling outlines of noticeable laser energy. This approach provides a low implementation cost, but it is not an appropriate low-cost explanation for radio-frequency identification (RFID) because if one delay is consistently greater than another across chips, then the response will always be the same.

Other authors, e.g., (51), used POWF in their work to decrease the cost of implementation but it reduces the performance.

2.4.2.2 Computational primitives

In computational primitives, the system security relies on the complexity in term of computation. These approaches provide complicated mathematical solutions with high security level. In particular, asymmetric cryptographic primitives have been approved as a strong solution to protect the data or secure the communication since every primitive has been considered a secure entity (52). In asymmetric cryptography, computational primitives exploit the basics of modular arithmetic to attain Shannon's philosophies of confusion and diffusion (53) with expending basic mathematical and logical processes. In (54; 55; 56; 57; 58), hash functions and symmetric cryptography schemes are presented. The main shortfalls of these

approaches are the use of large keys, high complexity and energy consumption assigned to the computation process. In addition, these approaches have not been investigated by public examination to experiment the security aspect.

2.4.2.3 Ultra Lightweight Protocols

Ultra lightweight refers to those security approaches that use simple logic operations such as exclusive-or (XOR) for their application (59; 60). These kinds of techniques become popular and more adaptive through RFID technologies. Indeed, their security aspect can provide the main services of the security resistant. These schemes focus on a set of applied assumptions, where the attacker will not have the facility to compute and generate the key. However, some of them consider that the attacker can read the tag for a too limited time slot. Ultra lightweight primitives can be classified into one-time pads, as in (61; 62), which are based on pseudonyms, re-encryption (58) and configured passwords (63).

As a result, secret keys are no longer essential to be securely stored in the memory of physical primitives. These techniques usually reduce the cost of implementation, as well as the security performance. The focus of computational primitives on complicated mathematical solutions makes them suffer from a security problem, in particular from key moving between the sender and receiver. A well designed technique may render them more popular and adaptive for low-end connected devices.

2.4.3 Modern Cryptosystems

Cryptography techniques contain certain limits, such as data modification by third parties who store them. These third parties must possess the customer data in the clear before modifying it, which poses a problem of data confidentiality and customer privacy. The homomorphic properties shown in Figure 2.5 allow the manipulation of encrypted data using arithmetic operations while sharing the information, guaranteeing that only the owner of the required private keys can decrypt the message and get hold of the contents. Figure 2.6 shows the homomorphic encryption technique against information interception threats in an untrusted environment. A homomorphic schema ensures data confidentiality for a cloud computing service. In other words, one of the main advantages of homomorphic encryption is delegating the calculations to cloud servers without losing data confidentiality. There are four significant types of homomorphic encryption, with the difference being in which supported mathematical operations and the periodicity at which these operations can be performed.

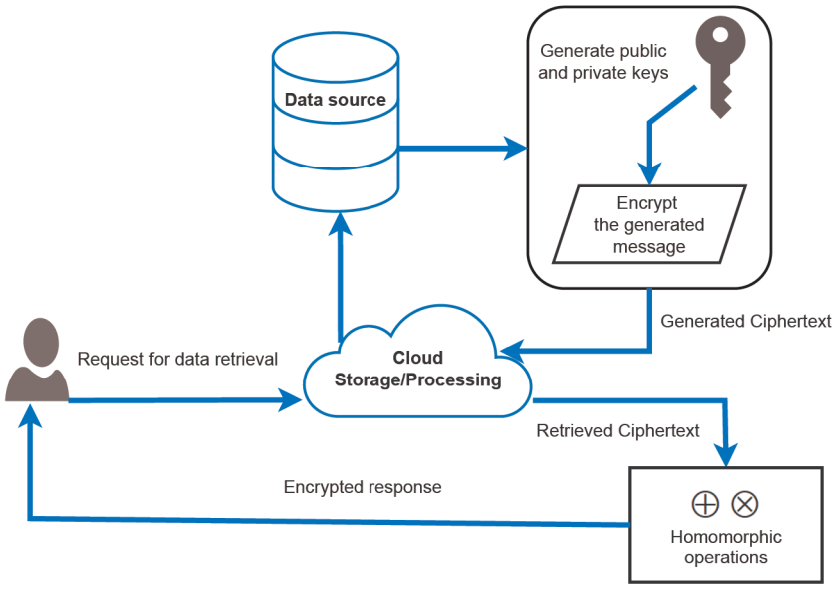


Figure 2.5: Homomorphic properties

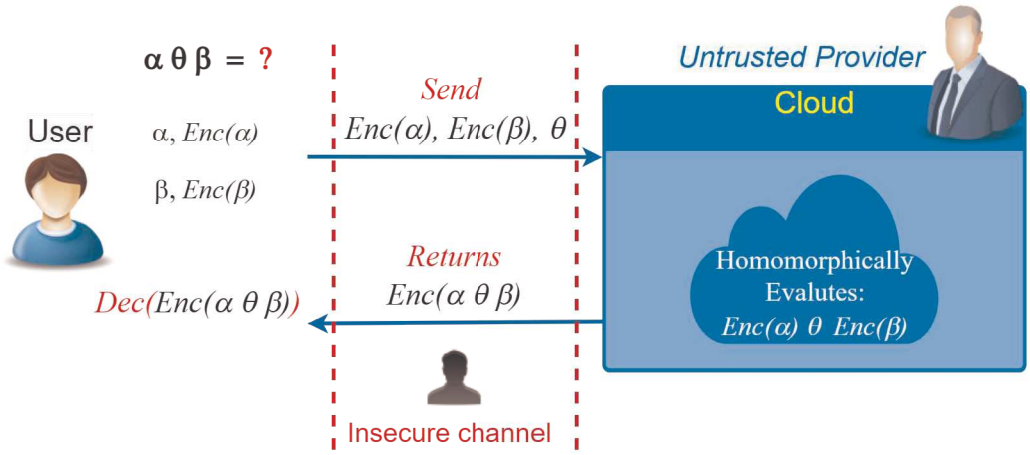


Figure 2.6: Homomorphic encryption scenario in an untrusted environment

2.4.3.1 Partially homomorphic cryptosystems

The first type of homomorphic encryption is called Partially Homomorphic Encryption (PHE). With this particular homomorphic encryption, specific mathematical operations can be carried out on the encrypted values that are obtained from the source.only one sort of operations, multiplication or addition, with an unlimited number of times (64). As multiplicative schemes, The RSA (43) scheme is a variant of PHE used to build SSL and TLS and secure connections. Other significant schemes include Paillier (65) encryption, which supports addition operations, and El-Gamal (66) encryption, which uses multiplication. As an additive cryptosystems, we can cite Naccache and Stern 1998 (67), Galbraith 2002 (68) and Kawachi et al. 2007 (69).

2.4.3.2 Leveled homomorphic cryptosystems

Leveled homomorphic encryption schemes (LHE) are also known as "leveled fully homomorphic," They allow for a limited number of addition and multiplication operations. Indeed, LHE supports finite operations over ciphertexts mainly from the perspective of circuit depth. The circuit depth will be predetermined in the setup algorithm (70; 71; 72).

2.4.3.3 Somewhat homomorphic cryptosystems

Somewhat Homomorphic Encryption (SHE) is a scheme in which certain operations like addition or multiplication can be performed to a certain complexity and The number of operations that can be performed are limited. This means that in LHE we can perform the addition operation for two ciphertexts and then multiply the result by another ciphertext. On the contrary, the SHE scheme allows us to do several additions or several multiplications without merging them (73; 74; 75).

2.4.3.4 Fully homomorphic cryptosystems

Fully Homomorphic Encryption (FHE) is newest scheme of homomorphic encryption, The FHE scheme can support both addition and multiplication operation. The initial idea of FHE was introduced by Craig Gentry in 2009 (76). FHE performs an absolute combination of operations on ciphertexts for an unlimited number of times (74). FHE can improve and secure current cloud computing systems and technology. (77; 78; 79; 80).

2.5 Conclusion

This chapter discusses several basic concepts related to the security of Internet of Things networks and the methods and techniques used in encryption systems to secure and protect the information exchange. We started by presenting concepts related to the architecture of IoT networks and the protection of connected smart devices. We also discussed the security challenges to securing smart devices and the currently adopted protocol stack to build Internet of Things networks and ensure the security and privacy of the information on the network. Next, we discussed encryption techniques and scheme types in terms of symmetry. Through this, we presented the four types of encryption systems and discussed security protocols and the essential element in securing distributed systems: the "key exchange protocol." Then, we showed how the cloud computing environment can provide a wide range of computing services and how they can be benefited from in the Internet of Things environment to ensure privacy and protect sensitive information, as well as modern encryption systems in the cloud computing environment, which depend on homomorphic encryption and its most important types.

Particularly in the context of these solutions, the security and protection of sensitive information remain significant concerns in distributed, cloud computing, and IoT environments. In particular, due to the weaknesses of interconnected devices in terms of power consumption, computation, storage capacity, and information communications. In the next chapter, we will address these concerns and provide a proposition for an encryption technique to protect the information in the Internet of Things environment. This technology is suitable for resource-constrained devices and networks. It offers a low encryption time and defends against common attacks, such as replay attacks, manipulation, destruction, or stealing sensitive information.

Chapter 3

A Flexible Encryption Technique for the Internet of Things Environment

This chapter presents a new scalable encryption technique, called Flexible encryption Technique (FlexenTech), to protect IoT data during storage and in transit. FlexenTech is suitable for resource constrained devices and networks. It offers a low encryption time, defends against common attacks such as replay attacks.

The work is published on Ad Hoc Networks journal (81).

3.1 Introduction

The Internet of Things (IoT) is a set of perceptible connected devices capable of interacting with each other. IoT devices embrace physical objects via digital actuators, RFID tags, sensors and communication units. IoT devices are often characterized by ultra-low bandwidth and limited computation and communication capabilities. As shown in Figure 3.1, most of the IoT scenarios are looking to connect with everything and everyone in order to share information. With the huge number of connected entities the network traffic as well as the storage capacity will increase in an exponential way. This introduces critical security challenges since efficient encryption, e.g., public key infrastructure, will not operate reliably. These limitations render IoT devices prone to a wide range of attacks. Cloud computing offers a solution to overcome the resource constraints of IoT devices. Furthermore, offloading data and outsourcing computational tasks to the cloud presents new security and privacy concerns. The amount of private information generated by IoT devices intensifies the security and privacy challenges in cloud-enabled IoT networks. As the adoption of IoT grows, the cost-benefit of designing rigorous protocols will become a major research subject with great impact. Hence,

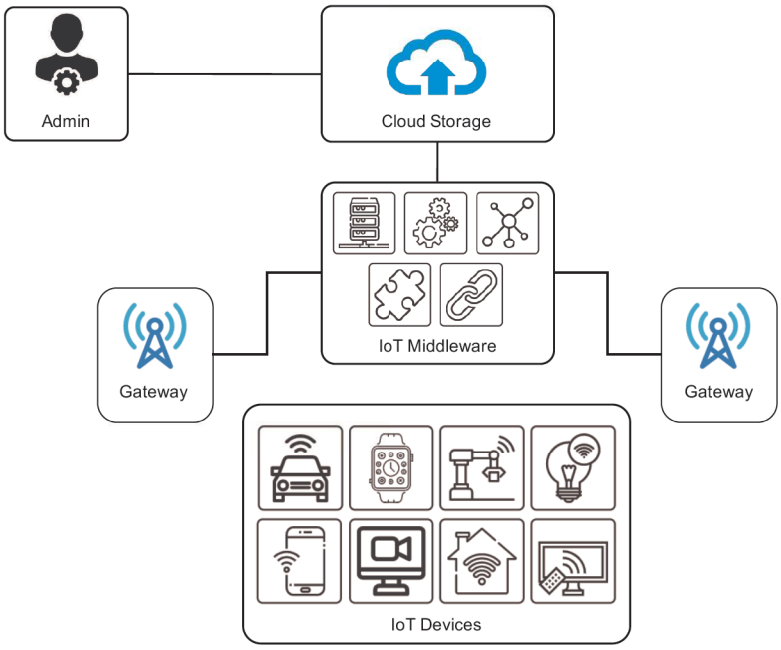


Figure 3.1: A comprehensive vision of the IoT architecture.

IoT applications cope with many challenges, particularly the difficulties related to privacy and security issues (82). These challenges include low resource availability issues, vulnerability problems due to the nature of wireless communication, and low-power computation resources. These constraints explain the difficulty of developing the necessary protocols for the management and protection of these networks, such as routing and security protocols. Like the other types of network architectures, security is one of the most challenging requirements in IoT networks (83) which includes mainly data confidentiality, data integrity, authentication, availability, and data freshness. Any adapted IoT architecture and its standard protocols must consider essential restrictions such as dynamicity, reliability, scalability, etc., and the information exchanged between IoT devices and the cloud must be encrypted with adequate security prior to transmission.

Most of the classical cryptographic systems and algorithms are not suitable for IoT networks as they are designed for resource-rich devices on high bandwidth networks. To address this issue, we propose a lightweight block cipher that consumes fewer resources and reduces encryption overhead by realising: smaller block size, smaller key length, lower computational complexity and simple key generation. Such a cipher depends on the specifically used hardware technology and coding style which are useful to provide an accurate IoT application. The critical issues that must be addressed in this cipher are: how to secure inter-device and device-to-cloud communications, how to ensure the security and confidentiality of various network entities, and what are the techniques adopted for authentication, access control and other security features?

3.2 Related works

Numerous research works have been recently carried out to provide encryption techniques pertinent to the IoT context. They address the inherent resource limitations of IoT devices that make it difficult to apply conventional encryption algorithms that require a significant amount of resources for their operation. Several researchers cope with the resource limitation challenge by proposing lightweight encryption techniques to provide efficient and secure communication with reasonable resource utilisation. A number of researchers devised techniques to optimise existing conventional block ciphers for IoT devices. Some of them have improved the performance of block ciphers by utilising smaller key sizes (84), reducing the number of encryption rounds (85) and using smaller block sizes (86). Other researchers focused on key management and authentication challenges in IoT environments (87). The vast majority of this research focuses on symmetric-based algorithms. For example, Beaulieu et al. (88) proposed SIMON and SPECK which are lightweight block ciphers with a variety of block and key sizes. The work in (89) proposed a model which uses parallel computation to enhance the performance of the block cipher algorithm. This model has been designed to consume lower energy and reduce hardware complexity. The SIT algorithm proposed in (90) focuses on a hybrid Feistel and substitution-permutation network (SPN) architecture. It is a mixture that uses the advantages of both approaches to develop a lightweight algorithm that provides substantial security while maintaining the computation complexity at a reasonable level in an IoT environment. The symmetric encryption scheme proposed by (91) uses multiple chaotic dynamical systems for the IoT domain. This technique maintains the dynamic key update with each input data block to provide higher levels of randomness. The authors in (92) present an experimental configuration of a fractional-chaos based-cryptosystem for an IoT-based architecture in ad hoc networks under the IEEE 802.15.4 standard. The solution benefits from the characteristics of the fractional-order derivative operator for encryption schemes to provide secure communications.

Lightweight asymmetric based solutions may provide stronger security than symmetric ciphers. However, asymmetric solutions are not highly scalable and often have higher computation complexity which renders them unsuitable for IoT environments (91). Recently, several studies such as identity-based encryption (IBE) (93; 94) have been published which combine user's identity with series of attributes to encrypt data while enforcing a secure access policy. Furthermore, the decryption is carried out when authorized users with the desired attributes satisfy a threshold access control policy. In (59), the authors proposed a lightweight IBE scheme by using pre-computation techniques to reduce the computation cost of constrained devices. They designed a lookup table, which obtains a pre-computed set of pairs generated using elliptic curve cryptography in order to be used later to carry out cryptographic operations at low computational cost. To overcome the

expensive bilinear pairing computations, the authors in (60) proposed a lightweight no-pairing IBE scheme based on elliptic curve cryptography by replacing pairing operations with point scalar multiplication.

In (95), the authors proposed a lightweight encryption scheme for smart homes. This scheme provides users and smart objects a secure service at the lowest computation and communication costs. The proposed solution relies on identity-based encryption to support flexible public key management, which does not require complex certificate handling.

Attribute-Based Encryption (ABE) provides the necessary flexibility required to manage a large number of IoT devices and fine-grained access control, which is very much suited to secure data transmission, storage and sharing in cloud-based IoT systems. The authors in (59; 96) give background information on the Ciphertext-Policy Attribute-Based Encryption and pre-computation techniques in an efficient and secure way, and then describe the extended ABE scheme proposed for the CloudIoT platform (97; 96).

In (98), Alphand et al. propose a Blockchain Security Architecture for the Internet of Things named *IoTChain*. In this work, the secure authorized access to IoT resources is ensured by a combination of the ACE authorization framework (99) and the OSCAR architecture (100). IoTChain replaces the single ACE authorization server by the blockchain to make the ACE authorization phase flexible and trustless. The feasibility of IoTChain is evaluated through an implemented authorization blockchain on top of a private Ethereum network. Dinu et al. (101) introduced a software benchmarking framework for the consistent evaluation of lightweight block ciphers on three different platforms, namely 32-bit ARM, 16-bit MSP430 and 8-bit AVR. Recently, a Novel Tiny Symmetric encryption Algorithm (NTSA) is presented in (102). NTSA proposes to enhance the text file transfer security through the IoT network by introducing additional key confusions dynamically for each round of encryption.

3.3 Summary and contribution

As a conclusion, all these works might provide a satisfying solution to secure the communication. However, there are still some inherent limitations in terms of efficiency, scalability and access control, which make them not suitable for constrained IoT devices. Therefore, there has been a growing demand for efficient lightweight encryption mechanisms that combine all the features of lightweight symmetric and asymmetric algorithms (91). Furthermore, these mechanisms should be able to provide mutual lightweight authentication for secure access control and authentication between users and devices in an IoT environment. In this paper, we

will design and implement a lightweight encryption technique, which is targeted to define a secure information exchange by considering the encountered limitations. To this end, an autonomous and secure exchange of confidential information in the IoT context is required in regard to the nature of the connected entities and their real environment. The proposed technique focuses on distributed architecture networks with the aim to ensure a high security on the basis of a dynamic topology.

3.4 Architectural concept

In this section, we present the main requirements for a secure information exchange that considers the limitations of IoT devices. Then we describe the proposed architecture that addresses these requirements. Our work focuses on distributed architecture networks with the aim to ensure an adapted security level with dynamic architecture topology. The reason behind flat and distributed network architectures is to adapt to the nature of IoT networks, where several entities of connected devices may autonomously and securely exchange confidential information.

A sustainable IoT architecture cannot be achievable without protecting the information, either during the transmission or the storage. The main tasks here are protection and management. Figure 3.2 shows a secure IoT management stack that contains two sub-stacks, IoT having a layered communication protocol as shown in the right sub-stack, that allows an enormous number of objects to get connected through the internet. These layers are to fulfill the IoT network limitations and needs. Suitable activity scenarios could be reached when the second sub-stack is accompanied by the first. That explains why the security paradigm of IoT network architectures becomes one of the most indispensable elements. Therefore, lightweight computation modules, such as simple permutation functions and bit-wise exclusive-or operation, are required in the design of secure transmission for each proposed protocol. To this end, we propose in this work a flexible encryption technique, called FlexenTech, that provides an acceptable security level with respect to IoT limitations and requirements.

Table 3.1 contains the abbreviations used in this paper together with a brief explanation to speed up the reading and ease the understanding of its content.

The proposed technique was designed with the following goals in mind:

- the technique should be a symmetric block cipher. The same secret cryptographic key (key = K) is used for encryption and for decryption. The plain data and cipher data are fixed-length bit sequences (block size = B).

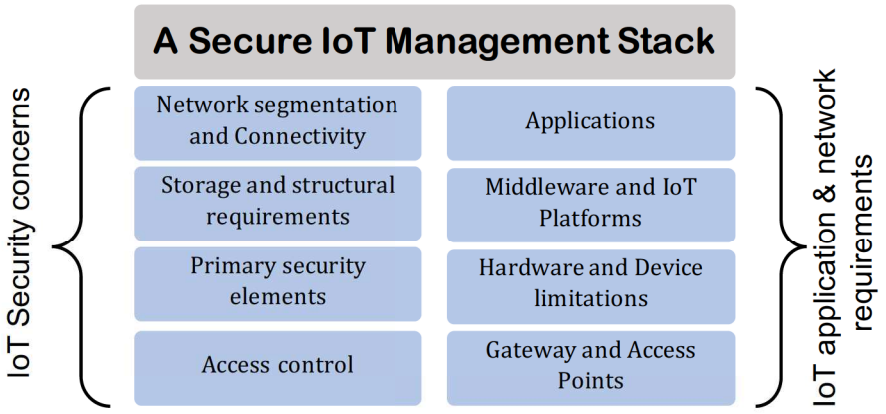


Figure 3.2: Secure protocol stack for IoT.

Acronym	Definition
K	is the secret **K**ey, each pair of entities asymmetrically shares the same secret number called K.
B	plain or ciphered data **B**lock size in term of bits.
V_i	represents the calculated **V**alue for the $\mathbf{i}^{th}$ bit position.
rnd	is the number of rounds that will be used in a given configuration.
r	the current round.
st_r	means the reversing start bit position from which the next bits will automatically be reversed.

Table 3.1: List of used acronyms

- it should be suitable for low resource hardware. This means that the proposed technique should use only primitive computational operations.
- it should be adaptable to devices of different data-lengths. Accordingly, the number B of bits in a data is a parameter of our proposed technique; different choices of this parameter can be used.
- it should be iterative in structure, with a variable number of rounds. The number of rounds is a second parameter as shown in our architecture (Figure 3.3). The user can explicitly maintain the trade-off between high speed and high security.
- it should be simple and easy to implement. It is more important to use a simple structure as illustrated in Algorithm 1, with the aim to easily implement it. Furthermore, a simple scheme is perhaps more interesting to analyze and evaluate, so that the cryptographic strength can be more rapidly determined.

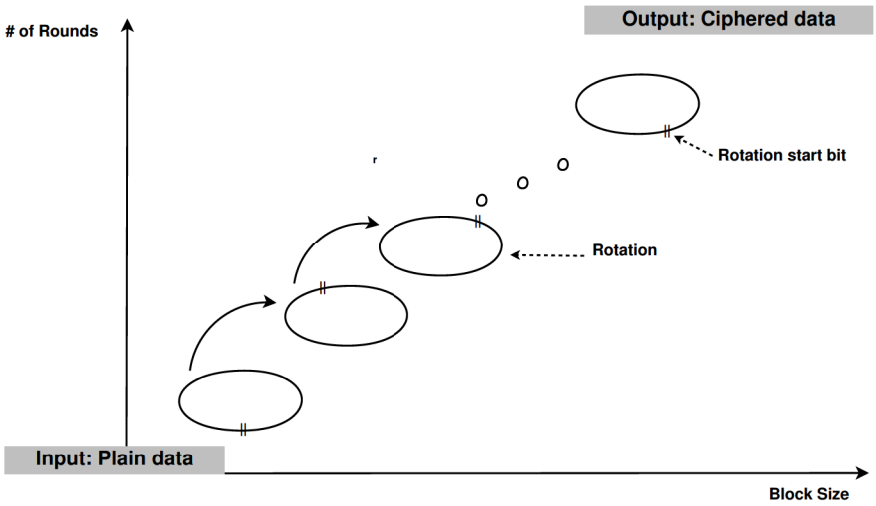


Figure 3.3: A conceptual representation of FlexenTech in view of the complexity space (*Block Size*, *Round*).

3.4.1 The proposed IoT encryption architecture

There are several emerging areas where highly constrained devices are interconnected to accomplish some tasks. These constrained devices (IoT) aim to communicate and make decision over many low resources and capacities. To an efficient use goal, the designed conceptual model illustrated in Figure 3.3 shows the complexity space of the proposed technique in view of the security requirements such as confidentiality, integrity and signature. The efficiency of FlexenTech resides on the choice of B and rnd, where its flexibility offers a complete freedom to the user for weighting between the complexity and the security level. In heterogeneous environments, the FlexenTech encryption mode can respond to many challenges and issues like power consumption of devices, limited battery, memory space, performance cost, and security.

The proposed technique focuses on three parameters to encrypt/decrypt the information:

- Each pair of nodes shares a secret number which will be used as a key.
- A number of rounds should be carried out before achieving the final encrypted information.
- In each given round i, a random rotation will be applied to the information obtained from the previous round $i - 1$.

The proposed architecture performs a set of random permutations, substitutions and rotations at the bit level to encrypt the plaintext. The permutations and substitutions are done using the modular function with the following steps:

- The size of the information to be encrypted is pre-configured with a given block size called B (in terms of number of bits).
- Using one of the proposed key management schemes, such as *Self-VKS* (103), the entities may securely share a given large number K. We note that B and K should verify the conditions gcd [1]$(K, B) = 1$ and $K > B$.
- The node uses Equation (3.1) to make a random permutation at the bit level of the plain information that will be sent. Figure 3.4 shows an example that explains how this equation produces random bit permutations.
- It is possible to construct lightweight hash functions with the aim to ensure a substitution based on local variable values. Since the modulus operation provides a new random i' for each given i, all bits between the st_r bit position and the B bit position will be reversed. Hence, this situation means that the content value of a given bit i will automatically be reversed if $st_r \leq i \leq B$. Otherwise, the content value will not be changed as shown in the example of Figure 3.6.

The key principle behind FlexenTech is that by securely sharing any given large number K, the entities can generate a set of B random values which will be used to encrypt and decrypt the exchanged information. The permutation of the bits is based on Equation (3.1).

$$V_i = (B \times (K - i)) \mod K \tag{3.1}$$

where $i = 1, 2 \ldots B$ and V_i represents the obtained value for the i^{th} bit position in the message of size B. Hereafter, the values of V_i will be ascendingly ordered with the aim to set up a random bit permutation. Every bit i will be permuted to its new position i' depending on the position of V_i in the ascendingly ordered list. To ensure that every bit will be permuted, the value of K should be selected in such a way that $\gcd(K, B) = 1$. Figure 3.4 shows an example of the proposed technique, where the entities securely share $K = 101$ and the size of the data is configured to $B = 16$. On the other hand, the destination node reconstructs the original information by applying Equation (3.1) on the received data to resubmit every bit on its right position.

[1] gcd(K,B) means the greatest common divisor of K and B

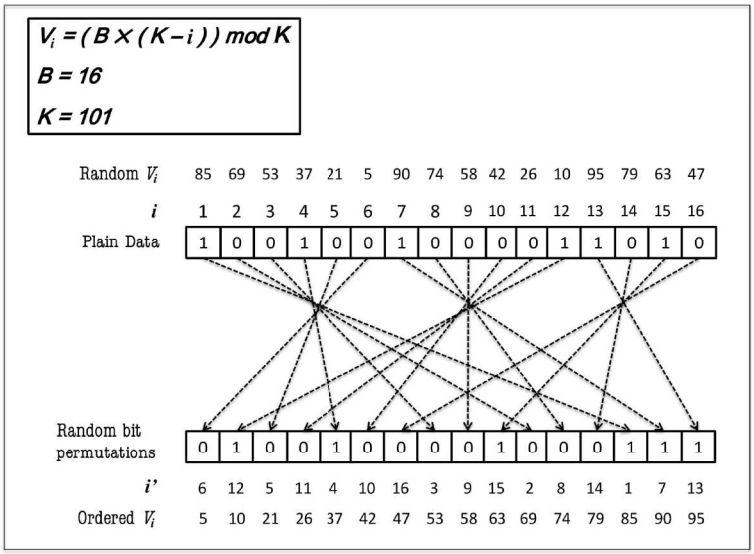


Figure 3.4: A demonstrative example of FlexenTech's permutation.

3.4.2 The proposed FlexenTech encryption algorithm with recursive rounds

The resource consumption of the proposed security protocol must be lightweight in terms of communication, memory usage and computation. To make sure that the proposed protocol is feasible and practical, it should add a communication overhead as small as possible and have low memory consumption and low CPU workload. Therefore, an efficient encryption mechanism has been adopted. The instructions for use of this mechanism are shown in Figure 3.5, where a set of rounds, performing the same operations over the obtained encrypted information, should be applied before obtaining the final encrypted information. On the other hand, the proposed technique focuses on a configurable security threshold which defines the preferred security level of the communication. The security threshold is the number of required rounds to cipher a plain data. Additionally, a given threshold determines the overload size that will be added in terms of the number of additional bits to ensure data integrity and signature. Furthermore, before maintaining any round, a random rotation is performed, where the start bit of the rotation is computed by Equation (3.2). On the other hand, the parameter number of rounds is considered as the security level because it implies immediately the similarity rate between plain and ciphered text, the computation time, and usually an additional overload.

$$st_r = (K \times (r + Nonce)) \mod B \tag{3.2}$$

where st_r is the rotation start bit of a given round r and $Nonce$ is a *nonce*, or could be a counter, which is used with the aim to reduce playback attacks. Figure 3.6

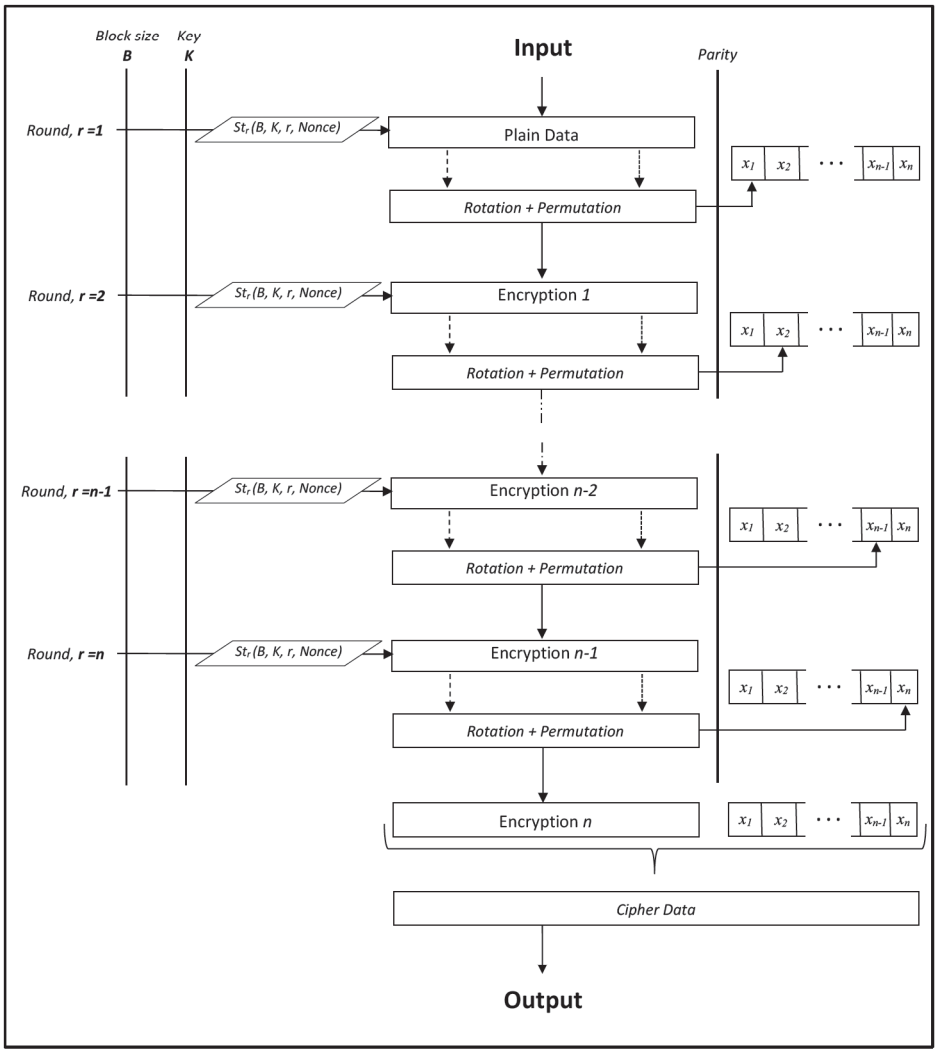
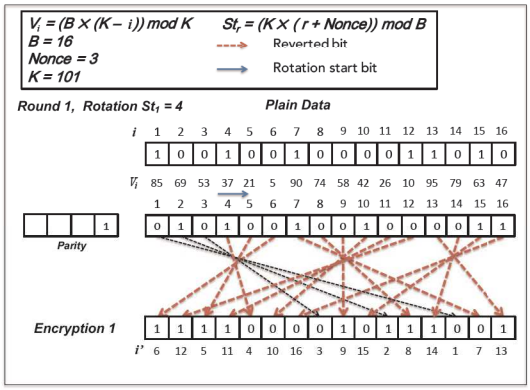


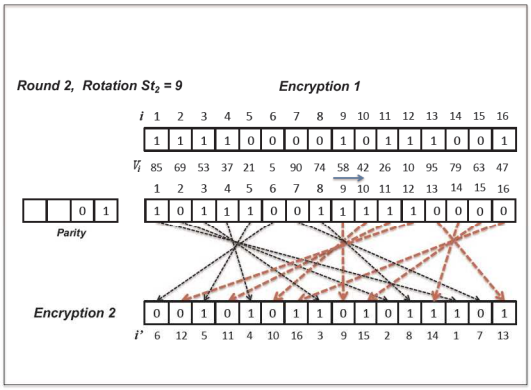
Figure 3.5: General structure of FlexenTech.

shows a clear example of the ciphering steps with a security level $rnd = 4$.

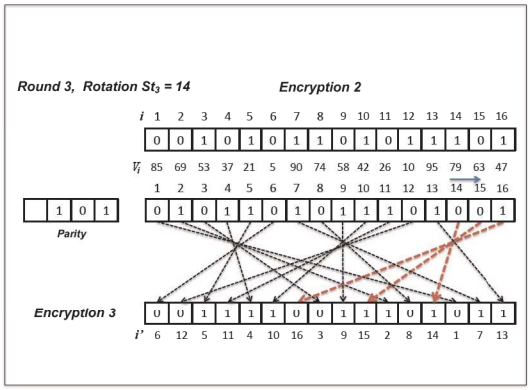
In the following, we explain how the proposed algorithm is applied within an IoT environment. For descriptive purposes, Algorithm 1 uses some terms defined in modular computation. An originator device prepares messages with the use of this algorithm in order to protect the data. The private data are the value of K and the secret information. The value of the data size, the used $Nonce$ and the security level, i.e., the number of rounds, can be used as public variables. The encryption table is defined to ensure a random permutation. Its size equals $3 \times B$, where the first row contains the values of V_i. The second and third rows are to save the original and the new bit position, respectively, after sorting V_i values. For more details, *FlexenTech.c 1* shows the C code of the example illustrated in



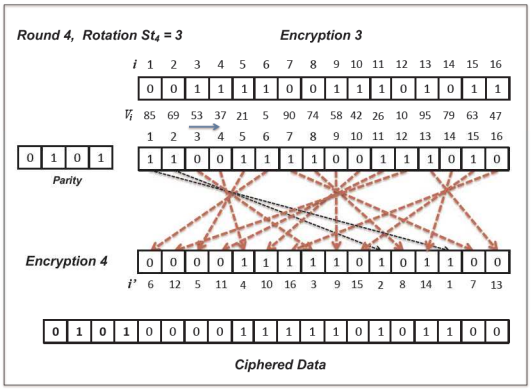
(a) Pre-configuration and steps to obtain the first round.



(b) Steps to obtain the second round.



(c) Steps to obtain the third round.



(d) Steps to obtain the fourth round and the final ciphered data.

Figure 3.6: A FlexenTech encryption example with four rounds.

Figure 3.6.

3.4.3 Analysis of the proposed encryption algorithm

The efficiency of any cryptographic protocol focusing on any algebraic structure depends on factors such as: parameter size, time-memory tradeoffs, available processing power, software and/or hardware optimization, and mathematical algorithms. Therefore, when designing the proposed technique we have taken into account the primary concerns to use powerful and efficient mathematical operations and algorithms which quickly carry out computations respecting at the same time the occurred overhead and the used memory storage. Several hash and encryption functions or digital signature schemes require computations in $\mathbb{Z}_m$, the integers *modulo* m (m is a large positive integer which may or may not be a prime). $\mathbb{Z}_m$ is exploited in many aspects of modern applied cryptography. In this section, we

Algorithm 1 Encryption and Decryption FlexenTech algorithm.

Require: Private: K, $plainData$;
Require: Public: B, $Nonce$, rnd, $mode$, $cipheredData$;
Ensure: $data$; ▷ $data$ `here means` $plainData$ `or` $cipheredData$ `which is tied to the encryption mode.`

Temporal: $table[3][B]$;
Constant: $V = 0$, $orgPos = 1$, $newPos = 2$;
for $(i \in [0, B-1])$ **do**
    $table[V][i] \leftarrow (B \times (K - i)) \mod K$;
    $table[orgPos][i] \leftarrow i$ ▷ `numbering the bit positions`
end for
$table \leftarrow sortingTable(table)$
if $mode = encrypt$ **then**
    $cipheredData \leftarrow plainData$
    for $r \in [1, rnd]$ **do**
        $st_r \leftarrow (K \times (r + Nonce))\ mod\ B$
        $cipheredData \leftarrow rightRotation(cipheredData,\ st_r)$;
        $cipheredData \leftarrow reverse(cipheredData,\ st_r)$;
        $parityBit(cipheredData,\ st_r)$;
        $cipheredData \leftarrow RandomPermutation(table, cipheredData)$;
    end for
    return $cipheredData$;
end if
if mode = decrypt **then**
    $plainData \leftarrow cipheredData$
    for $r \in [1, rnd]$ **do**
        $plainData \leftarrow RandomPermutation(table, plainData)$;
        $st_r \leftarrow (K \times (r + Nonce))\ mod\ B$
        $plainData \leftarrow reverse(plainData,\ st_r)$;
        $parityBit(plainData,\ st_r)$;
        $plainData \leftarrow leftRotation(plainData,\ st_r)$;
    end for
    return $plainData$;
end if
return -1

Algorithm 2 Sorting table function.

Require: (**Arg** *table*);
Ensure: *table*;
1: **Temporal:** *temp*;
2: **for** $(i \in [0, B-1])$ **do** ▷ `Sorting` *table* `depending on` V_i `values and moving their bit positions`
3: **for** $j \in [i+1, B-1]$ **do**
4: **if** (table[V][i] > table[V][j]) **then**
5: $temp \leftarrow table[V][i]$;
6: $table[V][i] \leftarrow table[V][j]$;
7: $table[V][j] \leftarrow temp$;
8: $temp \leftarrow table[orgPos][i]$;
9: $table[orgPos][i] \leftarrow table[orgPos][j]$;
10: $table[orgPos][j] \leftarrow temp$;
11: **end if**
12: **end for**
13: **end for**
14: **for** $(i \in [0, B-1])$ **do** ▷ `inserting the new bit positions`
15: **for** $j \in [0, B-1]$ **do**
16: **if** table[newPos][j] = i **then**
17: $table[newPos][i] \leftarrow j$;
18: **end if**
19: **end for**
20: **end for**
21: **return** *table*

discuss the requirements and the complexity of the proposed technique. We also discuss efficient methods that can be used to perform addition and multiplication in $\mathbb{Z}_m$.

As the designed technique is bit-oriented, all of the basic computational operations have $B - bit$ as inputs and outputs. Hence, the technique is a block-cipher with a one-word of B bit input ($plainData$) block size and a one-word ($cipherData$) output block size. The choice for B is not limited to such values. Therefore, the technique is well-defined for any $B > 0$, although for strong encryption and more efficiency it is proposed to use a large value of B. We note that there is no limit for the values of B and K (104), but the relation between them should verify $\gcd(K, B) = 1$ to ensure that

$$B \times i \ mod \ K \neq 0 \ \forall \ i \in \{1, 2, \ldots B\} \tag{3.3}$$

Lemma 1. *For $i \in \{1, \ldots B\}$ Let $V_i \in Z_k$ with $K > 0$. If B is relatively prime to K, then the congruence $B \times i \equiv V_i \pmod{n}$ has a solution i; moreover, any integer i' is a solution if and only if $i \equiv i' \pmod{K}$.*

Proof. The integer $i = V_i \times B'$, where B' is a multiplicative inverse of a modulo K, is clearly a solution. For any integer i', we have $B \times i' \equiv V_i \pmod{K}$ if and

only if $B \times i' \equiv B \times i \pmod{K}$ which holds if and only if $i \equiv i' \pmod{K}$.

Suppose that $B, V_i, K \in Z$ with $K > 0, B \neq 0$, and $gcd(B, K) = 1$. This theorem says that there exists unique integers i and j satisfying: $B \times i \equiv V_i \pmod{K}$, $B \times j \equiv V_j \pmod{K}$, and $V_i \neq V_j$ if $i \neq j$ with $0 \leq i, j < K$. In the proposed technique we avoid the value $i = 0$ to ensure a full bit permutation. □

We observe that any value of i that verifies $(B \times i) < K$ the congruence modulo K equals to $B \times i$ (i.e., $B \times i \equiv B \times i \; mod \; K$). This situation leads to generate some ordered V_i values. Following to the principle of the proposed technique, ordered values of V_i means encrypting data with weak permutation. For example, in Figure 3.4 V_i will be $[16, 32, 48, 64, 80, 96, 11, 27, 43, 59, 75, 91, 6, 22, 38]$ when we use Equation 3.3. The values that are in ascendant order (here there are three waves of ordered values: [16 – 96], [11 – 91] and [6 – 38]) will minimize the randomization of the permutation phase of the encryption process. In order to avoid these values, we let run i from $K-1$ until $K-B$ as shown in Equation (3.1). V_i values will then determine the bit permutation of plain data as shown by the example presented in Figure 3.4

An analysis shows that the total number of ordered compositions of V_i that can be obtained by using different values of K equals $B!$. Then, in the aim to reduce brute force attacks, the size of the data to be encrypted should be configured in such way that $B! \approx +\infty$. As a result, the strength of the proposed technique is relatively proportional to the size of B in terms of bits, where there is no restriction concerning the preferred size of K. The computation task of checking $\gcd(K, B) = 1$ can be avoided if the used size of B is a prime number, case in which any value of K will verify $\gcd(K, B) = 1$. Choosing a prime number B will help us to minimize the computation and to avoid repute computation in case that $\gcd(K, B) \neq 1$. The process of sharing securely the same value of K between two or more devices can be reached by using one of the known algorithms, for instance the asymmetric key sharing algorithm or a pre-configured symmetric key sharing scheme. On the other hand, the number rnd of rounds is the second parameter. Choosing a large number of rounds presumably provides an increased level of security. We note here that the number of rounds depends on the desired security level and the supplied size of B. However, choosing a large number of rounds also implies a need for more memory and computation time. The analysis of the example shown in Figure 3.6 says that a brute force attack may find the original data from the used permutation at a maximum of 16! tests. The speed of growth of the factorial function offers a great advantage to reduce brute force attacks.

In the proposed technique we use two kinds of permutation: the first is by ordering the values of V_i and the second is achieved from the rotation. However, the technique still needs a substitution function to avoid frequency analysis attacks.

Therefore, we have reversed all bits starting from st_r until the B-bit position. Hence, the value of st_r can be exploited for multiple uses: it offers a random permutation in each round, ensures the bit substitutions and extracts the integrity bits. The integrity is ensured by computing the parity of bits before starting the permutation. The size of bits parity equals the number of rounds. In this case, the receiver detects any change on one bit by comparing the parity bits of the obtained plain data with the right bit of the integrity bits. Otherwise, changing even bits will not be detected in the obtained plain data. This change can be detected in the next rounds. The integrity is ensured by satisfying the condition of preventing the bits change in case that an even number of changed bits are still in the same part, i.e., in the reversed bits part or in the other part. Hence, the number of rounds has a straight impact on data integrity strength.

Let P be the probability to make a change on some bits over the ciphered data that generate the same parity bits. The value of P is related to the size of parity bits, i.e., the number of rounds, and the value of $Nonce$.

In this technique, an attacker can falsify the data without any change at the parity bits level only when the falsified bits meet the following two conditions:

1. Falsifying an even number of bits situated in the reversed part, i.e., between $[start_r, B]$;
2. Each round contains an even number of falsified bits in the reversed part.

The first condition seems easy to be met, but each pair of falsified bits should be situated on the same side (either on the reversed or on the unreversed bits side) for every round. Hence, the second condition says that the data integrity of FlexenTech depends on the size of the parity bits, i.e., the number of rounds. Then, the probability P of an attacker to falsify e bits (e being an even number) of the ciphered data without a change at the parity bits is calculated as:

$$P^e_{st_r} = \frac{st_r}{B} \times \frac{st_r - 1}{B-1} \times \ldots \frac{st_r - e - 1}{B - e - 1} + \frac{B - st_r}{B} \times \frac{B - st_r - 1}{B-1} \cdots \times \frac{B - st_r - e - 1}{B - e - 1} \tag{3.4}$$

with total probability $P = P^e_1 \times P^e_2 \ldots P^e_{rnd}$.

Equation 3.4 says that there are two probabilities of successful falsifications. The first probability could be obtained when the falsification has been made over the unreversed side and these falsified bits are still on the same side for each round. The same holds for the second probability, but the falsification has been made over the reversed bits side. Hence, we observe that the number of rounds has an important impact on the probability to succeed in falsifying the ciphered data.

3.5 Discussion

We divide the discussion into two parts: 1) Choosing the adequate key establishment algorithm to securely compute the shared key (i.e., K). 2) The robustness and efficiency of this technique to cope with the most known attacks.

3.5.1 Key establishment algorithm

Key establishment is the most fundamental cryptographic primitive in all types of applications. However, the nature of such connected devices limits the use of conventional key establishment techniques. Many researchers have proposed to use symmetric key pre-distribution schemes for the key establishment objective (103; 105). These schemes consider the resource limitation, the communication capability and the computation speed (106). However, symmetric schemes often have a centralized aspect, focus on pre-configuration and define a pool key principle which leads us to exploit asymmetric key establishment schemes (107). The authors of (108) benchmark some public key protocols that are usually used for a symmetric key agreement to determine the most appropriate for the requirements of critical infrastructure and emergency applications. Based on these algorithms, the analysis shows that there are remarkable performance benefits with the $Curve25519$, especially $Four\mathbb{Q}$ (109) which uses the *Elliptic Curve Diffie-Hellman* scheme.

In the proposed protocol, we use $Four\mathbb{Q}$ which is a new elliptic curve algorithm released by Microsoft Research in 2015. $Four\mathbb{Q}$ is not used yet in standard or known protocols. Technically, $Four\mathbb{Q}$ targets the 128 bit security level and its high performance is mainly obtained from the decomposition of the total number of the elliptic curve group operations and fast arithmetic modulo computation using the Mersenne prime $p = 2^{127} - 1$. In term of speed, $Four\mathbb{Q}$ uses the endomorphisms to accelerate scalar multiplications via four-dimensional decompositions, where the computation of scalar multiplications is significantly faster than all known curve-based cryptographic primitives.

3.5.2 FlexenTech's robustness and efficiency

During the development of the FlexenTech encryption technique, we began to focus our attention on how to find adequate solutions to the encountered IoT issues by taking into account the designed technique that should additionally provide a high-security level when suitable parameters are assigned. Our technique demonstrates

its robustness and efficiency by providing data encryption, data integrity and data freshness, as will be explained below.

3.5.2.1 Encryption

The proposed technique converts a plain text into a ciphered text with the same text size B. Furthermore, the proposed technique considers permutation and substitution to avoid frequency cryptanalysis attacks. That means that the obtained ciphered text has no trace or information related to the original text. In computation and efficiency terms, the proposed technique needs only simple operations in order to encrypt a text with a given key K by computing B values of V_i and ordering them. In each round, the bits of the input text should be rotated starting from st_r and ordered matched with V_i values and reversing the bits that are on the right side of st_r, see Figure 3.6(a). The key restriction here is that the value of K should be greater than B which offers a large pool of keys and ensures flexible key configurations. This encryption mechanism offers a pool of size $B! - B$.

3.5.2.2 Data integrity

The integrity approaches aim to detect the alteration in the data that will be sent. Any even slight modification should be of effect on the integrity bits. In the decryption phase, the authorized entity has to check a possible alternation by computing the parity bit of the bits that are listed on the right side of st_r, then compare it with the originally received bit in each round. This technique is powerful for detecting the integrity of data during the data decryption, i.e., we do not need to execute an additional separated task. Furthermore, it is possible to detect a data alteration before obtaining the plain text. In fact, this principle will offer a significant reduction in term of computation. On the other hand, such applications request to hide the integrity bits in the ciphered data which is easily reached in our proposed technique by computing the r next V_i values (V_{r+1}, $V_{r+2} \dots V_{r+B}$) under the condition that $K > (r + B)$. The parity bits and the ciphered data can then be encrypted in an additional round.

3.5.2.3 Data freshness

The importance of data freshness is increasing, especially in the domain of distributed smart networks which are composed of a large-scale set of autonomous data sources. The use of the encrypted data with same freshness may lead to unauthorized access problems. In fact, data freshness is indispensable in Wireless Sensor Networks, IoT and other domains, to reduce various types of attacks

such as replay attacks and extraction of some confidential information from the encrypted data. At the same time, we have to be interested in the overall generated overhead in order to ensure data freshness. In this paper, we propose an efficient technique to ensure the data freshness by using a *Nonce* during the data encryption/decryption phases, where the used *Nonce* has no restriction in term of size. The way to avoid the augmentation of the overhead is to introduce the *Nonce* to compute the rotation in each round. In fact, by using this principle the following services can be ensured:

- The Data freshness is ensured without any additional overhead. However, the entities publicly share the used *Nonce*.
- Encapsulating the whole encrypted data, that will be sent, allows to distinguish ciphered text from the same plain data by using a different *Nonce* value in each case.
- A large counter size is useful to be used as a *Nonce* in order to encrypt data with different counter values, i.e., for each secret K the counter can be used 2^x times, where x is the *Nonce* size in terms of bits.

The data freshness offers many other services and advantages mainly in real-time applications including e-commerce, sensor data fusion, traffic control, and monitoring. To reach this aim, the data freshness process of this technique focuses on the communication speed, where no overhead is added. The proposed technique gives a configurable range of parameter values so that user devices may run the encryption algorithm whose security and speed are accepted for their application. Unlike several encryption techniques, which have no parameterization and hence no flexibility, the proposed encryption algorithm permits upgrades as necessary. The choice of r affects both encryption speed and security. For some domains, real-time may be the most critical requirement. This kind of application looks for the best security obtainable within a given encryption time requirement. Choosing a small value of rnd (say $rnd = 4$) may provide some security, albeit modest, within the given speed constraint. The size overhead is critical, because it immediately affects the communication speed and the lifetime of connected devices, as well as the packet loss rate caused by packet collisions.

3.6 Implementation and results

In this section we will show the implementation of the proposed cryptographic algorithm to be analysed for its flexibility evaluation. In its encryption process, FlexenTech uses a pseudo random number as key for encryption and decryption. It

completes the *rnd* rounds of encryption on each B bits block of data. In all rounds, encryption is done using a function that ensures an irregular rotation. Further, each round increases the number of integrity bits. With any unfixed parameter, the encryption strength of FlexenTech is directly tied to the predefined parameters during its use. The main advantage of FlexenTech is that the user has the full choice for weighing between the lightweight encryption, security level, and data overload. Hence, FlexenTech is efficient for software that runs in smaller processors (smart cards) and embedding in hardware. It allows implementers to customize encryption speed, key setup time, and code size to balance performance.

We present some preliminary results on the strength of the proposed encryption technique. The encryption process of FlexenTech is simple and easy to implement efficiently in most known script languages. For the evaluation purpose, the proposed technique has been coded in *C* (*see the source code of FlexenTech*) and follows the description given in Section 3.4.2 to obtain the result shown in Figure 3.6. The analysis evaluates the effect of the parameters number of rounds (*rnd*) and block size (B) as a function of the time performance. Since the study relies on the development of a lightweight encryption technique for the Internet of Things, the proposed technique will be implemented and verified in two environments with the following specifications:

Raspberry Pi 3 Model B :

- CPU: Quad-Core 1.2GHz Broadcom BCM2837 64bit, ARM Cortex-A53 (ARMv8)
- RAM 1GB
- Raspbian is the official operating system
- Geany GCC c/c++ compiler

Laptop Acer Aspire E1-571 :

- CPU: Intel Core™ i5-3230M 3rd Gen, 2.60 GHz Dual-core, HM77 Express
- RAM 8GB
- Windows 10 operating system
- Geany GCC c/c++ compiler

Details of implementation and results are given below.

3.6.1 Number of Rounds Experiments

First, we start by investigating two experiments that show how changing the number of rounds may affect some properties of the proposed technique. Figures 3.7(a) and 3.7(c) show the influence of *rnd* on the running time in the personal computer and the Raspberry Pi device, respectively. The block size is fixed to 16 through all runs. As observed from Figure 3.7(a), when $rnd = 4$ the running time in the personal computer does not exceed 0.25 *ns*. Hereafter, when *rnd* is raised to 8, the running time is doubled to reach 0.5 *ns*. The same trend is observed with other tests. We conclude that the time increases approximately by a factor of two.

Regarding the influence of *rnd* on the running time in the Raspberry Pi device, according to Figure 3.7(c), it is clear that the tendency is the same. Indeed, the running time in the Raspberry Pi device continues to increase as *rnd* gets bigger.

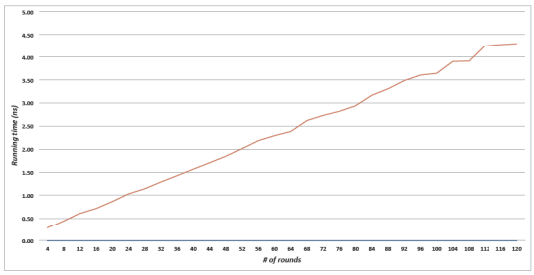
3.6.2 Block Size Experiments

The experiments illustrated in Figure 3.7(b) and 3.7(d) aim to extract the relation between the size of the used block with the computing time and the encryption complexity. For that reason, we fixed *rnd* to 16 and we tested six different block sizes: 16, 32, 64, 128, 256 and 512. Figure 3.7(b) shows that, when the block size is as small as 16 bits, the running time does not exceed 0.35 *ns*. When the block size is fixed to 32 and 64 *bits* the running time increases by a factor of 2 with a time of about 0.75 and 1.85 *ns*, respectively. When the block size equals 128 *bits* the running time does not increase as much and stands at 2.1 *ns*. However, in case the block size is up to 256 and 512 *bits*, the running time increases hugely, about 180%. In fact, this huge augmentation in the running time is caused by the undertaken time performed by the sorting function in Algorithm 1 (see line 9).

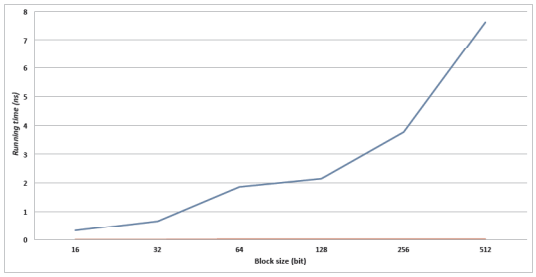
According to Figure 3.7(d), and as in the case of *rnd*, we also observe that the influence of the block size B on the running time in the Raspberry Pi device follows the same tendency as in the personal computer.

3.6.3 Comparative study

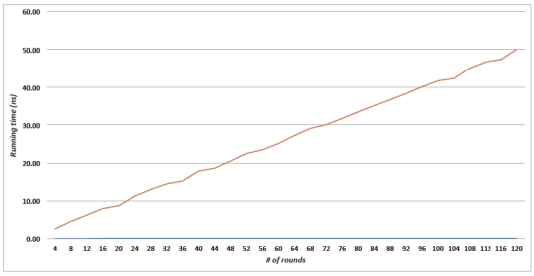
In this section we compare the encryption time and the flexibility of FlexenTech with the most known symmetric encryption techniques. Figure 3.8 shows the encryption speeds of FlexenTech and other symmetric techniques in a personal computer. We observe that the encryption speed of FlexenTech is better than the AES and Blowfish encryption techniques. The encryption processing time of RC5 and FlexenTech is almost identical. In fact, both are simple algorithms



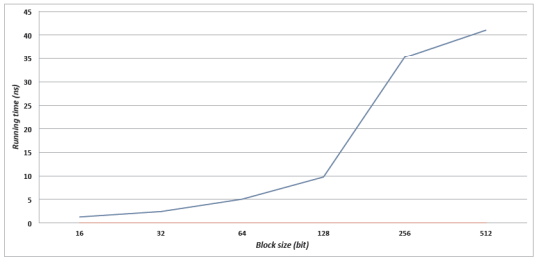
(a) Running time of the personal computer as a function of the number of rounds.



(b) Running time of the personal computer as a function of the block size.



(c) Running time of the Raspberry Pi device as a function of the number of rounds.



(d) Running time of the Raspberry Pi device as a function of the block size.

Figure 3.7: Running time as a function of the number of rounds and the block size.

which have low memory requirements and are suitable for hardware or software. Figures 3.9 and 3.10 show the encryption processing time executed on a Raspberry Pi device. We observe in Figure 3.9 that FlexenTech is fastest and also provides good security if the used block size is smaller than 64 bits, whereas RC5 provides security only if suitable parameters are chosen. The strength of RC5 depends on the size of the used key, which is strong possible if it is long. As a result, the proposed FlexenTech is efficient and more flexible in term of the block size, where we could use any block size. On the other hand, we have extracted the encryption processing time as a function of the number of rounds. Figure 3.10 confirms that RC5 is better if the used number of rounds is greater than 8. However, FlexenTech still gives a reasonable encryption processing time even if $rnd < 32$. Due to the random permutations at bit level of the plain data it is possible to use one round to encrypt the information. Therefore, four rounds are largely enough in FlexenTech, whereas twelve rounds are required in RC5.

In the side of flexibility purpose, Simon and Speck encryption schemes designed with leveled flexibility allowing some different key and block sizes. Speck encryption contains 10 variants where the block size is limited to five values (32, 48, 64, 96, 128) and the key size is limited to seven values (64, 72, 96, 128, 144, 192,

256) and The number of rounds depends on the parameters selected (nine values), it gets its nonlinearity from the modular addition operation, they noted that key lengths below 80 bits or so do not provide an especially high level of security. Whereas, when SIMON implemented on 8-bit AVR microcontroller, Speck encryption with 64-bit blocks and 128-bit key consumes 192 bytes of Flash, temporary variables consume 112 bytes of RAM, and takes 164 cycles to encrypt each byte in the block, the *Attacked Rounds/Total Rounds* percentage ranges between 53% and 74%, but reduced-round variants have been successfully attacked.

TEA encryption routine focuses on a Feistel iterations by using 64-bit size blocks (32 bit words) and a 128-bit key (the key will be divided into 4 parts) where odd rounds use K[0; 1] and even rounds use K[2; 3]. With the use of 32 cycles (64 rounds), the designers mention that sixteen cycles may suffice but they suggest 32 cycles with the constant "C" (used to prevent simple attacks based on the symmetry of the rounds). There are attacks on 17 rounds TEA because it admits several related-key attacks which arise from the severe simplicity of its key schedule (110) . We compare the performance of the proposed technique with algorithms shown in Table 3.3 to take a global view in terms of flexibility and performance. In general, the security performance in block cipher varies with the block size, the key size and number of rounds. Other encryption techniques prefer a large block size for faster execution, because, with larger block size leads to a large chunk of data will be encrypted in a single execution cycle. While with small block sizes, the same size of input data would require more execution cycles. Moreover, large keys result in slower computation time, because, all bits of the key are involved in an execution cycle of the encryption. Whereas, the proposed FlexenTech offers a major advantage at this view point. FlexenTech could offer efficient encryption using only one single round and it also ensures the encryption and integrity at the same time just with few number of rounds. Another gain in the complexity view, the key has two parts where the first is a static part which is used once for the random permutation for all rounds. The second is a dynamic part which requires its specific computation in each round to compute the start bit rotation. The static part offers a considerable gain in term of computation, resources occupation and energy consumption certainly for the low-end IoT devices.

3.7 Conclusion

Nowadays, the main security challenge in the IoT environment is the design of efficient and lightweight encryption techniques in regard to resource limitations of connected devices. In this paper, we have proposed the FlexenTech encryption technique to cope with the device limitation challenge in order to render these devices more applicable in terms of security requirements. The mechanism of the

Technique	Configuration			Performance		
	Block size	**Key size**	**# of rounds**	**Computation time**	**Energy consumption**	**Against attacks**
FlexenTech	Variable	Variable	Open	Speed with such configuration	Low energy consumption	Brute force; Replay attack
RC5	32,64,128 bits	0–2040 bits	12 rounds	Reasonable time	Low with small # of rounds (111)	Differential attack
TEA	64 bits	128 bits	6–64 rounds	Speed with small # of rounds	Low energy consumption	Equivalent key attack
SIMON	64–256 bits	32–128 bits	32–72 rounds	Suitable time (112)	Moderate energy consumption	Weak keys
Blowfish	64 bits	32–448 bits	16 rounds	Moderate time (113)	High energy consumption (111)	Known plaintext
DES	64 bits	56 bits	16 rounds	Relatively slow (113)	Low with small block	Brute Force Attack
AES	128 bits	128, 192, 256 bits	Tied to the key size (112)	Relatively slow	Relatively low with small block (114)	Known plaintext

Security	Standard encryption		Lightweight encryption			
	RC5	**Blowfish**	**FlexenTech**	**TEA** (115)	**SIT** (90)	**SIMON** (88)
Confidentiality	★★★★	★★★	★★★★	★★★	★★★	★★★
Authentication	★	★	★★	★★★	★★★	★★
Integrity	★★★	★★	★★★	★★	★★	★★
Authorization	★	★	★	★★	★	★★
Freshness	★★	★★	★★★	★	★★	★★

Table 3.3: Summary table of encryption techniques with various configurations

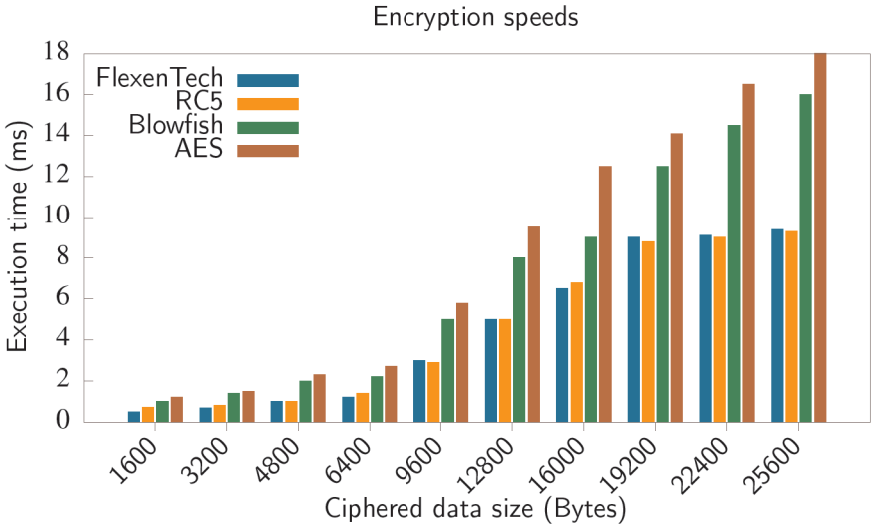


Figure 3.8: The execution speed of FlexenTech compared with other encryption techniques.

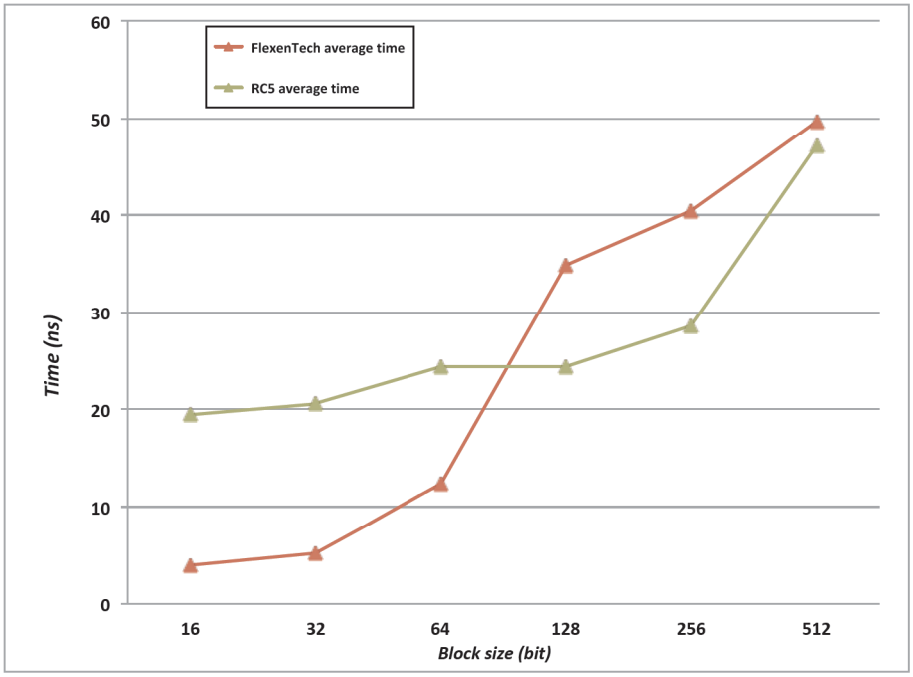


Figure 3.9: The execution time as a function of the block size.

proposed technique reduces the computation time incurred by data transmission, because we have minimized the number of rounds used to cipher the information and introduced many improvements to minimize computation. Reducing computation time offers a remarkable benefit in term of energy consumption of the autonomous IoT devices. The implementation of FlexenTech upon low-end IoT

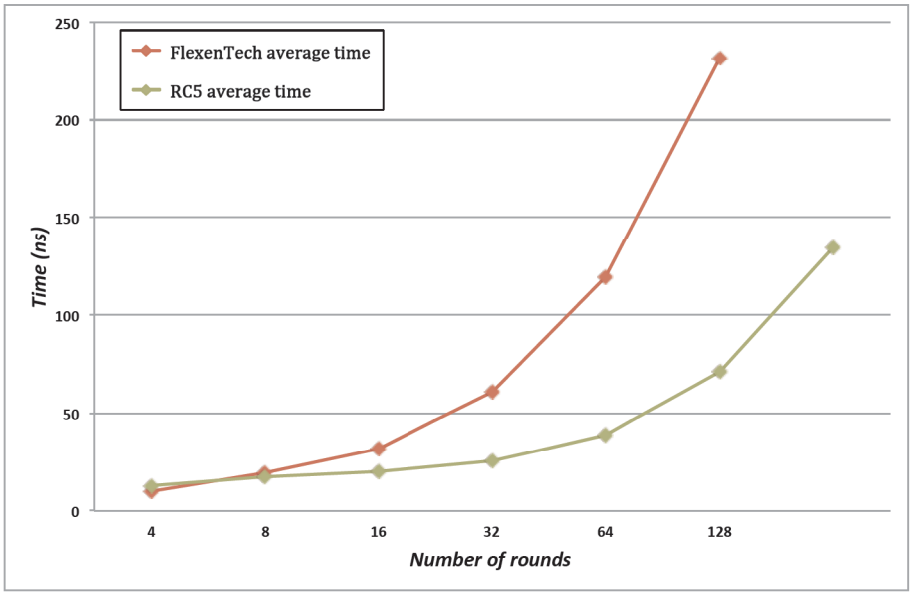


Figure 3.10: The execution time as a function of the number of rounds.

devices shows its practical flexibility. This flexibility enables the choice of a minimal number of rounds and a smaller overhead size which leads to an improvement in encryption time whilst not reducing security levels. The obtained enhancement on the encryption time decreases many risks of such attacks. A security analysis of the proposed technique also shows its robustness and feasibility.

In future research, we intend to investigate the possibility to adapt this principle to the use in cloud storage. Furthermore, with the rapid progression of the Internet of Things (IoT) and the emerging Big Data paradigm in the context of Cloud-enabled large-scale sensor networks, we plan to orient the FlexenTech technique to the Big Data environment.

Chapter 4

Homomorphic encryption technique to secure Cloud IoT environment - Collaborative work

This chapter results from collaborative work to develop encryption techniques based on modern cryptosystem that adopt fully homomorphic encryption in the cloud environment and enable it in the Internet of Things environment. The work has been applied in smart healthcare, where collects health data through IoT devices connected to patients to nurse or prevent critical situations. The initial proposal for this work was published at The 4th International Conference on Future Networks and Distributed Systems - St. Petersburg, Russian Federation, November 2020, entitled: "A Homomorphic Digit Fragmentation Encryption Scheme Based on the Polynomial Reconstruction Problem" (45). The final results are published in an article entitled "A Fully Homomorphic Encryption based on Magic Number Fragmentation and ElGamal Encryption: Smart Healthcare Use Case" in Expert Systems Journal, 2021 (44).

4.1 Introduction

The development of smart cities is based on the Internet of Things technology. IoT technology will continue to develop and have a significant impact on how we live.Cloud infrastructure today powers the majority of smart cities. Cloud computing has now become a necessity in information technology. Effective cryptographic techniques are needed to assure data confidentiality, integrity, availability, safety, and robustness to make cloud entities cyber-secure and autonomous. However, the recently discovered issue of confidentiality preservation has presented a severe challenge to computer scientists and mathematicians. Thus, over the last decade, security concerns about digital data and algorithms have increased exponentially

due to the growth of network communications and a significant increase in processing capabilities. However, various of attacks are frequently attempted against this mode of communication, such as changing, erasing, or stealing sensitive information.

To protect sensitive information against these attacks, legacy encryption systems use a shared encryption key between the peers involved in communication to encrypt the exchanged information (116). In many cryptosystems, even consumers or service providers have elite rights over the data by using the encryption key, especially for popular cloud services, where the privacy of sensitive data will be lost. Furthermore, The third party could share the encrypted data without accessing its content. Nowadays, preserving confidentiality has led researchers to cope with the homomorphic encryption challenges.

In cryptography, Homomorphic Encryption (HE) can be used to address these concerns by enabling any third party to apply computational functions on encrypted data without decrypting them beforehand. The idea of the HE scheme stems from the question, "Is there an encryption function (Enc) such that $Enc(m_1 + m_2)$ is easy to compute from $Enc(m_1)$ and $Enc(m_2)$?" Basically, the question points to exploring whether any algebraic homomorphic encryption scheme can be designed. Indeed, HE relates to a mapping in the abstract algebra (117), which is structure-preserving for two algebraic structures of the same type (such as two groups, two rings, or two vector spaces) (117).

To increase the level of security in the cloud by utilizing these features, making it independent in the calculation, and proposing new encryption schemes.

4.2 Related works

The initial idea of homomorphic encryption was introduced as a concept shortly after the development of the RSA cryptosystem in 1978 by Rivest et al. (43) with the appearance of asymmetric encryption (118). The RSA encryption considers only multiplicative operations Partial Homomorphic Encryption (PHE), for example, which either ensures multiplication (Equation 4.1) or addition (Equation 4.2) but not both. Hereafter, several homomorphic encryption schemes have appeared (45; 65).

$$Dec(Enc(m_1) \times Enc(m_2)) = m_1 \times m_2 \tag{4.1}$$

$$Dec(Enc(m_1) + Enc(m_2)) = m_1 + m_2 \tag{4.2}$$

Proposed in 1999 by Paillier, the Paillier cryptosystem allows homomorphic addition from $(\mathbb{Z}^*_{n^2}, \times)$ to $(\mathbb{Z}_n, +)$.

An other example of PHE is GM's scheme (119). Goldwasser-Micali proposed the first probabilistic public-key encryption scheme, which is an additive system. that accomplishes the additive equation (Equation 4.2). The hardness of GM comes from the quadratic residuosity problem (QRP). A number α is called quadratic residue modulo n if there exists an integer k such that, $k^2 \mod n = \alpha$ and the QRP decides whether a given number q is quadratic modulo n or not. If $m = m_1 m_2 \dots m_i$ and $c = c_1 c_2 \dots c_i$, the GM encryption could be described as follows :

$$c_i = Enc(m_i) = (y^2 \times x^{m_i}) \mod n, \ \forall m_i \in \{0, 1\} \tag{4.3}$$

where x (public) is a quadratic nonresidue modulo n, and y_i is a quadratic nonresidue produced such that $gcd(y_i, n) = 1$. In the decryption, if $(c \mod p)$ is a quadratic residue the function returns 1, otherwise 0.

In (65), the authors proposed an additive homomorphic cryptosystem from $(\mathbb{Z}^*_{n^2}, \times)$ to $(\mathbb{Z}_n, +)$. Recently, the explosion of generated data and the unavailability of sufficient resources (in terms of computation or storage) has lead researchers to face the Fully Homomorphic Encryption (FHE) issue.

Boneh et al. (120) have created a scheme allowing unlimited additions and one multiplication (Somewhat HE). The paper has introduced a public key homomorphic encryption scheme which creates the encryption $\psi(x_1, ., x_n)$ of the variables $x_1, ., x_n$, where ψ is a 2-DNF formula on boolean variables $x_1, ., x_n \in \{0, 1\}$. The proposed homomorphic scheme as follows:

$$c = Enc(m) = g^m \times h^r \mod n \tag{4.4}$$

where, g , h and u are the generators $(h = u^q)$, r is a random number $\in \{0, n-1\}$, using the secret key p, the decryption function $Dec(c) = log_g(c)$, $c = c^p$ and $g = g^p$. The hardness of BGN's scheme comes from the hardness of the subgroup decision problem which consists of deciding whether a given element is a member of a subgroup Gp of group G of the composite order $n = p \times q$, where p and q are distinct primes.

In (121), the authors have introduced a threshold fully homomorphic encryption scheme (TFHE) from the learning with errors (LWE) assumption. A universal thresholdizer is constructed from the proposed TFHE, which is used to resolve the main issue of single-round threshold signatures from lattices. In addition, threshold functionality can be added to numerous systems, such as CCA-secure public-key encryption (PKE), signature systems, and pseudorandom functions.

Many SWHE schemes contain noise that prevents performing an arbitrary degree of operations. With the aim to reduce the noise, several methods are applied after

each operation. In 2009, bootstrapping has been used by Gentry (76) which is based on ideal lattices. Bootstrapping offers unlimited additive and multiplicative homomorphic operations (Full HE) which is obtained from the reduction of noises in each operation. Van Dijk et al. proposed a simple fully homomorphic encryption (FHE) scheme in (79) which is represented as follows:

$$c = Enc(m) = m + 2 \times r + p \times q \tag{4.5}$$

where $m \in \{0, 1\}$ and $r << p$, the decryption algorithm is $m = Dec(c) = (c \mod p) \mod 2$, and the hardness of the technique comes from the hardness of the Approximate-Greatest Common Divisor (AGCD) problem.

The proposed scheme is of conceptual simplicity compared to Gentry's technique. The proposed bootstrappable encryption scheme uses addition and multiplication over the integers instead of using ideal lattices over a polynomial ring. However, the cost of this simplicity is a too large public key size $O(\lambda^{10})$. Fan et al. (122) have ported Brakerski's fully homomorphic scheme based on the Learning With Errors (LWE) problem to the ring-LWE (RLWE) setting. The tight worst-case bounds have been derived from the noise caused by numerous homomorphic operations like multiplication, relinearisation, and bootstrapping. A modulus switching trick has been used to simplify the bootstrapping analysis. A fully homomorphic with a certain level of security has been achieved through deriving concrete parameters.

In order to minimize noise, a batching technique has been used. Despite the recent development of several FHE schemes, unfortunately, none of them shows its feasibility with respect to the massive amount of generated noise which increases the computational complexity (78). Gentry and Halevi (123) have pioneered in implementation of an FHE scheme in 2011. However, this scheme implementation outcome was unsatisfied as it took a long time around 900 seconds to add up two 32-bit integers and around 67,000 seconds to multiply them. In addition, it requires more than 780,000 bits of ciphertext to encrypt a single bit. The public-key size ranges from 70 Megabytes to 2.3 Gigabytes based on the setting size. To run one bootstrapping operation (on a 1-CPU 64-bit machine with large memory), the time ranges from 30 seconds to 30 minutes based on the setting size. Yagoub et al. (124) proposed an adaptive and efficient fully homomorphic encryption technique:

$$c_i = (m_i + rand_i \times k) \mod (k \times p) \tag{4.6}$$

where $rand_i = (m_i \times k) \mod p$ and $f^{-1} : m_i = c_i mod k$. To provide an order-preserving scheme, Yagoub et al. have used a simple linear expression of the form $a \times x + b$. In order to mask the value of the used integers, the coefficient b is secret and the value of $a \times x$ will be hidden in b by using the addition operator. Linear expressions must respect the indexing order which leads to define the following order preserving function: $index_i = p \times m_i + rand_i$.

In (125), the authors proposed a novel homomorphic encryption scheme for integer arithmetic. The proposed Dyer et al. scheme focuses upon the Chinese Remainder Theorem (CRT) secret sharing, where the security of their scheme comes from the assumption of the toughness of the partial approximate common divisor problem (PACDP).

In Doröz et al. (126), the authors presented a homomorphic evaluation of the prince block cipher. Their leveled implementation is based on a generalization of NTRU. In KeyGen function, a decreasing sequence of primes $q_0 > q_1 > \cdots > q_d$ is used with a polynomial $\Phi(x) = x^n + 1$. For each i, $u^{(i)}$ and $g^{(i)}$ is sampled from the distribution D, set $f^{(i)} = 2 \times u^{(i)} + 1$ and $h^{(i)} = 2 \times g^{(i)} \times (f^{(i)})^{-1}$ in ring $R_{q_i} = \mathbb{Z}_{q_i}\big[x\big]/\big\langle\Phi(x)\big\rangle$ (If $f^{(i)}$ were not invertible, it could not be easily re-sampled). They sample for $i = \big\{0, ..., d\big\}$ and for $\tau = \big\{0, ..., \big\lfloor logq_i \big\rfloor\big\}$, $(s_\tau)^{(i)}$ and $(e_\tau)^{(i)}$ from D, the publish evaluation key $ek = h^{(i)}(s_\tau)^{(i)} + 2(e_\tau)^{(i)} + 2^\tau(f^{(i-1)})^2$. To encrypt a bit $b \in \big\{0, 1\big\}$ with a public key $(h^{(0)}, q_0)$, the following equation is used:

$$c^{(0)} = h^{(0)} \times s + 2 \times e + b \tag{4.7}$$

where s and e are random from D. To decrypt the ciphertext c with its corresponding private key $f^{(i)}$, the decryption maintains multiplying the ciphertext and the private key in R_{q_i} and then computes the message by modulo two: $m = c^{(i)} \times f^{(i)} \mod 2$.

In (127), the authors relied on El-Gamal as an encryption technique, where they divided the ciphertext into three parts, of which the first two were almost identical to El-Gamal. The first part α^k is random for each message and it will be used later to decode the two other parts. The second part is $m \times \beta^k$ knowing that $\alpha^p = \beta$, and the third part was devoted to the calculation of the addition of two numbers, $c_3 = \beta^{k+m}$. The fundamental difference between this work and our's is that we only use two values to represent the ciphertext so that to calculate the addition, we use the linear representation, which is faster to decode. Decoding $\beta^{m_1+m_2}$ to get $m_1 + m_2$ takes a very long time (discrete logarithm) and affects the size of m. For this, they noted in their work that m must be small in size, unlike our technique where $m < k - 1$. We have noticed several gaps in this work the most important of which is that we put $\alpha^p = \beta$ with $pk = (\alpha, \beta)$, knowing that $\alpha^p = \alpha + r \times p \rightarrow pk = (\alpha, \alpha + r \times p)$. It is obvious that it is simple to calculate $r \times p$ where p is considered a secret key, and therefore more correct to use $y \neq p$ where $\alpha^y = \beta$ so that the $pk = (\alpha, \beta)$ will be protected.

Several cryptosystems like RSA and Modified Paillier are deterministic encryption schemes, i.e., if we encrypt the same message more than once, we will always get the same ciphertext. Formally, if $c \neq c' \Rightarrow m \neq m'$ where $c = Enc(m)$ and $c' = Enc(m')$. Deterministic encryptions do not satisfy IND-CPA (Indistinguishability under Chosen Plaintext Attack) security, in other words, they are vulnerable to

CPA and we can say that they are not semantically secure. In CPA, the attacker having c and he tries to find m where $c = Enc(m)$, he chooses m' and calculates $c' = Enc(m')$. After that, he decrypts $c \times c'$ to obtain $m \times m'$ and find m.

Deterministic schemes do not satisfy FTG (Find-then-guess) security. In FTG, the attacker makes encryption requests to the Oracle, which in turn returns the encrypted one of these two messages. The attacker can make other requests to the Oracle, but must eventually guess what message the Oracle encrypted. A scheme is said to be FTG if the attacker cannot easily distinguish which of the two messages was encrypted. Therefore, a deterministic scheme cannot verify this level of security because the attacker has the right to request the encryption of the two messages he has chosen. In our system, we have avoided this type of attack by proposing a probabilistic encryption scheme where we can encrypt the same message in a lot of different ways. Formally, we can find m and m' where $m = m'$ and $c \neq c'$.

In order to improve cryptosystems efficiency by increasing speed (reducing computation time) and reducing data size and complexity, we have developed several encryption techniques that we will explain in the next point.

4.3 Proposed techniques

First technique
In (44), we propose an FH encryption scheme that aims to provide a secure and robust scheme with an infinite number of homomorphic additions and multiplications. The proposed homomorphic technique could be designed in a demonstrative cloud computing architecture as shown by Figure 4.1, by which the ciphertext is divided into two parts. The first part is a linear encryption scheme that aims to ensure the homomorphic addition property. On the other hand, the first linear part will be used as a parameter for the encryption/decryption of the second part which is to ensure the multiplicative property.

The homomorphic additive property focuses on factorization and fragmentation problems, whereas the multiplicative homomorphic propriety exploits the El-Gamal encryption principle with specifications. In which, the generator g is a number independent of public key y (in classical El-Gamal $y = g^k$ where k is a secret key). Also, we do not have a random number generated for each message to be encrypted (in classical El-Gamal $c = (m \times y^r, g^r)$ where r is a random number, in MNF-G $c = m \times g^m$). So, the power of g equals m (m is the original message to be encrypted which is a decryption of the first part). The second part depends on the discrete logarithm issue. Furthermore, the use of the El-Gamal technique (i.e., computation of g^m) leads us to speed up the power computation of large integers.

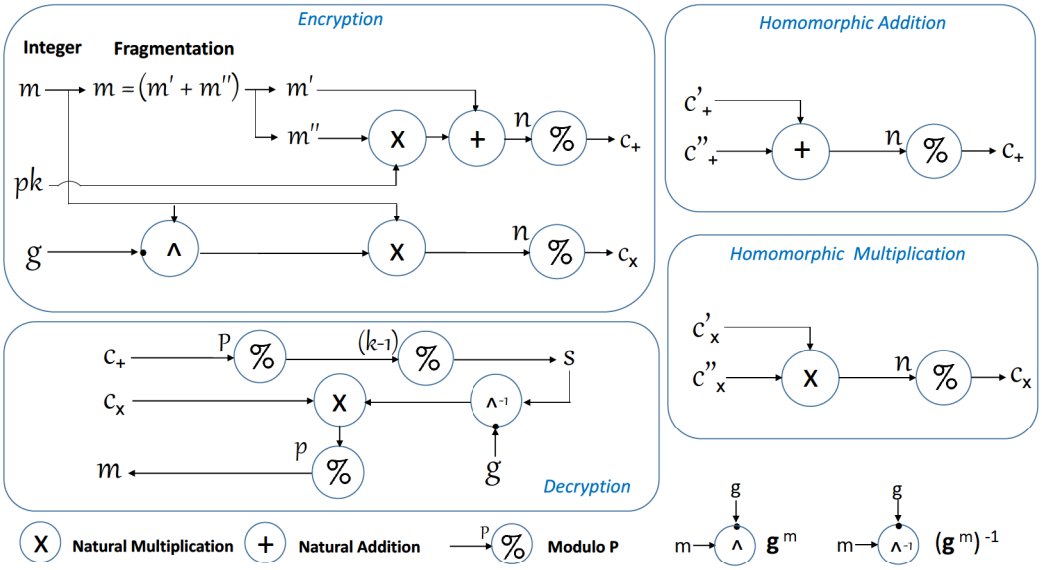


Figure 4.1: Homomorphic property

The obtained results are demonstrated in Subsection 4.4.2. RSA also could provide homomorphic multiplication. However, we have focused on El-Gamal since it provides message integrity property. For example, after any given operation with RSA, $c+c' = (m+m', (m \times m')^k)$ where $m = m_1 \times k + m_2$ and $m' = m'_1 \times k + m'_2$; with El-Gamal, $c+c' = (m+m', (m \times m') \times g^{m+m'})$. We observe that in El-Gamal $m + m'$ appears on both sides. Therefore, the decryption of the first part will be used to decipher the second part.

In the literature, we meet several symmetric FHE schemes that are efficient in terms of resources allocations and speeding. However, these schemes suffer from some critical concerns. The key sharing issue is one of the met concerns since the pair should use a private canal to share the symmetric key. Asymmetric FHE provides more robustness comparatively with symmetric schemes. The complexity and resources allocation of asymmetric encryption is the major issue. The proposed MNF-G scheme is to reduce the complexity of an asymmetric scheme by using two encryption fields. However, several proposed asymmetric encryption techniques take a very long encryption time due to their exponential complex computation. We focus on how to provide a linear cyclic asymmetric encryption that will lightweight and accelerate the encryption time and maintain strong and secure encryption against attacks. We start the explanation of the proposed FH scheme with the following linear cyclic encryption.

$$c = (m + r \times p) \mod n \tag{4.8}$$

Where $n = p \times q$. The weakness of Equation (4.8) is prone to a known plain text

attack. To cope with this weakness and other kind of attacks, we introduce to Equation (4.8) some improvements as follows

$$c = (m \times k + r \times p) \mod n \tag{4.9}$$

In fact, this is the origin of the El-Gamal encryption mode (66). Nonetheless, Equation (4.9) is still weak because when someone can get two plain texts, he will be able to recover p and then obtain all other ciphered entities. The core purpose of the proposed encryption scheme is to hide the plain text before using Equation 4.9 by randomly fragmenting m into two parts m' and m'' such that $m = m' + m''$ so that it becomes difficult to an attacker to find the used m' and m'' with the aim to extract p. Hence, the proposed encryption equation will be as follows:

$$c = (m' + m'' \times k + r \times p) \mod n \tag{4.10}$$

The principal modes of encryption, decryption, homomorphic addition, and multiplication are shown in Figure 4.1. Some challenges have appeared to verify the homomorphic encryption property because the multiplication of $c_1 \times c_2$ will produce k^2. To ensure the homomorphic multiplication we represent the ciphered text c in the form of two parts, c_+ and $c_\times$, where $c_\times$ is encrypted by using the El-Gamal principle.

The El-Gamal cryptosystem is an efficient cryptosystem which provides high level of secure communications (128). El-Gamal is an asymmetric key encryption algorithm with homomorphic properties for public-key cryptography which is based on the Diffie-Hellman key exchange invented by T.El-Gamal in 1985. Its scheme depends on the one way function, meaning that the encryption and decryption are done in separate functions. In our proposed scheme, the key generation algorithm is a finite cyclic group $\mathbb{Z}_p^*$, where p is a large prime, a generator g is generated, and a random number α with $1 < \alpha < p-1$ is selected. The integer private key α must be kept secret, and the corresponding public key is (g, β, p) such that, $\beta = g^\alpha \mod p$. For the encryption of a message m, a random number $k \in \{1, \ldots, p-1\}$ is selected for each message, and $Enc(m) = (X, Y)$ such that $X = g^k \mod p$ and $Y = (m \times \beta^k) \mod p$. The pair (X, Y) can be decrypted by $m = (Y \times ((X)^\alpha)^{-1}) \mod p$.

The security, in this case, is based on the complexity of solving the discrete logarithm problem modulo a large prime. For this the authors of (129) note that if the plaintext space is not large, then messages m_1 and m_2 could instead be encoded as g^{m_1} and g^{m_2} , respectively, and in this way $(X_1 \times X_2, Y_1 \times Y_2)$ could also be considered as a ciphertext of $(m_1 + m_2) \mod p$ encoded by $g^{m_1+m_2}$. Several variants of El-Gamal have appeared, such as the classical El-Gamal encryption scheme, the El-Gamal cryptosystem in Z_n (where n is not necessarily prime), the El-Gamal cryptosystem in the domain of Gaussian integers, the El-Gamal cryptosystem over

quotient rings of polynomials over finite fields and the bilinear El-Gamal encryption scheme which is used in a secure privacy-preserving data aggregation (130).

The core nature of the FHE scheme design
In the following, we will explore in-depth the MNF-G (Magic Number Fragmentation and El-Gamal encryption).

Consider the primitives P,C,K,$Enc()$, $Dec()$ and the cyclic group $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z} = \{0, 1, \ldots, p-1\}$, where

- P is the ring of plain text $\mathbb{Z}_p$
- C is the ring of encrypted text $\mathbb{Z}_n$
- K is the key ring
- $Enc()$ is the encryption function $Enc_{pk} : P \longrightarrow C$ with $pk \in K$
- $Dec()$ is the decryption function $Dec_{sk} : C \longrightarrow P$ with $sk \in K$

The proposed fully homomorphic encryption scheme consists of three processes:

- KeyGen: which returns a secret and public key sk and pk, respectively, $(sk,\ pk) = (k, k + r \times p, g)$
- $Enc(m)$: a plain text $m \in P$ is encrypted using the public key pk, $c_+ = (m' + m'' \times pk) \mod n$, $c_\times = m \times g^m) \mod n$
- $Dec(c)$: a ciphered text is decrypted using the secret key sk, $m = (c_+ \mod p) \mod (k-1), m = c_\times \times (g^m)^{-1} \mod p$

This proposal focuses on the hardness of the number factorization and the magic fragmentation of m. Table 4.1 shows the exchange scenario between owner, users and cloud, where $m < k - 1$. **Second technique**
In (45), we propose a LFHE technique which offers a high security level with consideration of the robustness. Therefore, we use Equation 4.11 to homomorphically encrypt the data:

$$\psi = (\pi + r \times p) \mod n \tag{4.11}$$

where $n = p \times q$, p and q are large prime numbers and r is a random number. This encryption is fully homomorphic $\forall \pi < p$. Unfortunately, the untrusted part may easily reveal p and decrypt any encrypted data. To improve this encryption scheme with a limited noise and to attempt to increase the depth of the multiplication, we decided to treat each digit of m apart instead of treating the whole m, i.e., each digit m_i is between 0 and 9 so $m_i \times k < m \times k'$ where $k << k'$. Hence, we

Time	Type	User	Cloud
t_0	Trapdoor k, p and q	random: g, $r > q$ prime : p, q, k public : $n = p \times q$	
t_1	Secret key	private: (k, p)	
t_2	public key	$(pk = k + r \times p, n, g)$	(pk, n, g)
t_3	Homo. key	n	n
t_4	Encryption	$c = ((m' + m'' \times pk) \mod n$ $(m \times g^m) \mod n$	$(c_+, c_\times)$ $(c_+, c_\times)$
t_5	Homo. computation		$(c_+ + c_+) \mod n$ $(c_\times \otimes c_\times) \mod n$
t_6	Decryption	$m = (c_+ \mod p) \mod (k-1)$ $m = c_\times \times (g^m)^{-1} \mod p$	$(c_+, c_\times)$ $(c_+, c_\times)$

Table 4.1: The homomorphic key sharing, encryption and decryption scenarios.

consider that m is written in a base k and then we convert it to base 10 (as seen in Section 3.2). In the decryption, we successively divide the cypher text c by k^i .

The proposed homomorphic encryption scheme consists of the following processes:

- KeyGen: it returns a secret and public key sk and pk, respectively, where $(sk,\ pk) = ((k, p), (k + r \times p, n))$
- $Enc(m)$: a plain text m is encrypted using the public key pk, $c = m_0 \times pk^0 + m_1 \times pk^1 + \ldots m_i \times pk^i$
- $Dec(c)$: a ciphered text is decrypted using the secret key sk, $m = \sum_{i=l}^{0}(\frac{c}{k^i}) \times 10^i, c \leftarrow c - c \times k^i$

This proposal focuses on the hardness of the number factorisation and the polynomial reconstruction problems such that $(\frac{c}{k^i})$ is the quotient of $c \div k^i$.

4.4 Performances

In the following, we will see some properties and performances of the first proposed techniques Magic Number Fragmentation and El-Gamal encryption (MNF-G).

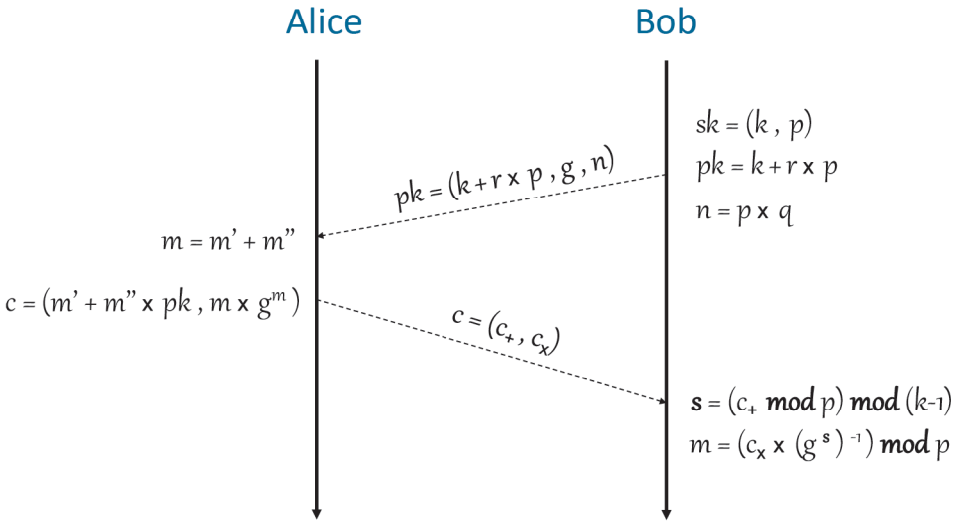


Figure 4.2: The encryption scheme

4.4.1 Magic Number Fragmentation and El-Gamal encryption (MNF-G)

Encryption
Asymmetric encryption of the proposed scheme allows Alice to send Bob a ciphered text using the shared public key which works as illustrated in Figure 4.2. With $pk = k + r \times p$ and the encryption of $c = (m' + m'' \times pk) \mod n$, i.e., $c = m' + m'' \times (k + r \times p) = m' + m'' \times k + r' \times p$. $m' < p \rightarrow c \mod p = m' + (m'' \times k) \mod p$ and $m'' < p \rightarrow (m'' \times k \times k^{-1}) \mod p = m''$

Decryption
$c = (c_+, c_\times)$ and $c_+ = m' + m'' \times k + r \times p$. In a first time, we extract m from c_+, since $m \times k < p$. Then $c_+ \mod p = m' + m'' \times k$ because $r \times p \mod p = 0$ and since $m \times k \mod (k-1) = m$ then $Dec(c_+) = m$. After that, we will exploit it to decipher the second part $c_\times$ by using the g, and if we note $s = Dec(c_+) = m$ so $m = (c_\times \times (g^s)^{-1}) \mod p$ **Homomorphic Additive property**
$Enc(m_1) + Enc(m_2) = m'_1 + m''_1 \times pk + m'_2 + m''_2 \times pk = (m'_1 + m'_2) + (m''_1 + m''_2) \times pk = E(m_1 + m_2)$. and $c = c_1 + c_2 + \cdots + c_i = (m'_1 + m''_1 \times pk) + (m'_2 + m''_2 \times pk) + \cdots + (m'_i + m''_i \times pk) = (m'_1 + m'_2 + \cdots + m'_i) + (m''_1 + m''_2 + \cdots + m''_i) \times pk = m' + m'' \times pk$, where $m' + m'' < p$. Hence $Dec(c) = m_1 + m_2 + \cdots + m_i = m$. In general, we obtain

$$\sum_{i=1}^{t} Enc(m_i) = Enc(\sum_{i=1}^{t} m_i) \tag{4.12}$$

Homomorphic multiplicative property
Let c_1 and c_2 be two ciphered texts encrypted from $m_1 = m'_1 + m''_1$ and $m_2 = m'_2 + m''_2$, respectively. $Enc(m_1) \times Enc(m_2) = (m_1 \times g^{m_1} \times m_2 \times g^{m_2}) = (m_1 \times m_2 \times g^{m_1} \times g^{m_2}) = (m_1 \times m_2 \times g^{m_1+m_2}) = (m \times g^s)$ where $s = m_1 + m_2$. Noting that $s = Dec(c_1 + c_2)$, in order to guarantee the property of the multiplication, it is necessary to guarantee the decryption of $(m \times g^s)$, and that by using the addition of the first part. $Enc(m_1) \times Enc(m_2) = (c_{1+} + c_{2+}, m_1 \times m_2 \times g^{m_1+m_2})$. In general, we obtain

$$\prod_{i=1}^{t} Enc(m_i) = Enc(\prod_{i=1}^{t} m_i) \tag{4.13}$$

4.4.2 Key generation and power function

One of the drawbacks of the exponential function is the long computing time especially when low capacity devices are used. Proposing an efficient exponential computation method to overcome this dilemma will make our cryptosystem even stronger. The idea here is to convert the power into a multiplication, by calculating the vector vec based on the length of the input with respect to $\mod n$, that we denote by $len(m)$ for an input m, and then to calculate g^m using this vector. More precisely, let $m = m_0 m_1 m_2 \dots m_i = m_i \times 10^0 + m_{i-1} \times 10^1 + \dots + m_0 \times 10^i \Rightarrow g^m = (g^{10^0})^{m_i} \times (g^{10^1})^{m_{i-1}} \dots \times (g^{10^i})^{m_0}$. We just need to calculate $vec = [g^{10^0} \mod n, g^{10^1} \mod n, \dots, g^{10^{n-1}} \mod n]$ to obtain

$$g^m = \prod_{i=0}^{t} vec_i^{m_i},\ with\ 0\ <=\ m_i\ <=\ 9\ and\ t\ =\ len(m) \tag{4.14}$$

Table 4.2 presents a comparison between our $vec - power$ function and the power function pow of the python math library, using an HP Laptop with a Processor Intel(R) Core(TM) i3 and 4 Go RAM. In test 1: $len(m) = len(g) = len(n) = 1400\ digits$ and in test 2: $len(m) = len(g) = len(n) = 2100\ digits$ In the following,

$g^m \mod n$	test 1	test 2
$vec - power(m, vec, n)$	0.33 s	1.22 s
$pow(g, m, n)$	0.78 s	2.64 s

Table 4.2: Two power tests

we will see some properties and performances of the second proposed techniques A Homomorphic Digit Fragmentation Encryption Scheme Based on the Polynomial Reconstruction Problem (DFE-PR).

4.4.3 Digit Fragmentation Encryption Based on the Polynomial Reconstruction Problem (DFE-PR)

Encryption

Let $m = m_i m_{i-1} \dots m_0$ where m_0 is the weak point of m and $m_j \in \{0, .., 9\}$ $\forall j \in \{0, .., i\}$

$c = m_0 \times pk^0 + m_1 \times pk^1 + \dots m_i \times pk^i = m_0 \times k^0 + m_1 \times k^1 + \dots m_i \times k^i + r \times p$ noting that: $m_0 \times k^0 + m_1 \times k^1 + \dots m_i \times k^i < p$. This operation could be defined by the following algorithm :

Algorithm 3 Encryption algorithm

Require: m, pk, n
Ensure: $c = Enc(m)$

```
1: function ENC
2:     l ← length(m)
3:     c ← 0
4:     for i ← 0 to l do
5:         d ← string(m, [l − i − 1 : l − i])
6:         c ← (c + d × (pk^i)) mod n
7:     end for
8:     return c
9: end function
```

Decryption

Lemma 2. $(k-1) \times (k^0 + k^1 + \dots k^{i-1}) < k^i$

Proof. $(k-1) \times (k^0 + k^1 + \dots k^{i-1}) = k^1 + k^2 + \dots k^i - (k^0 + k^1 + \dots k^{i-1}) = k^i - k^0 = k^i - 1 < k^i$ □

While $m_i <= 9 \wedge 9 < k - 1 \Rightarrow 9 \times k^0 + 9 \times k^1 + \dots 9 \times k^{i-1} < k^i \Rightarrow \frac{c}{k^i} = m_i$ after that $c \leftarrow c - c \times k^i$ As things progress, we will obtain $m_0, m_1, \dots m_i$, then we calculate $c = m_0 \times 10^0 + m_1 \times 10^1 + \dots m_i \times 10^i$ and to ensure the recovery of m, we do the division on k^i where i is the length by digit of c, and $i > j$ where j is the length by digit of m because $k > 10$ ($\Rightarrow c > m$). Likewise, the decryption is correct for $m_i < k$ (Equation 2). The decryption operation is shown in the following algorithm :

Algorithm 4 Decryption algorithm

Require: c, k, p
Ensure: $m = Dec(c)$

function DEC
  $c \leftarrow c \mod p$
  $l \leftarrow length(c)$
  $m \leftarrow 0$
  for $i \leftarrow 0$ to $l + 1$ **do**
    $d \leftarrow \frac{c}{k^{l-i}}$
    $c \leftarrow c - d \times k^{l-i}$
    $m \leftarrow m + d \times 10^{l-i}$
  end for
  return m
end function

Homomorphic Additive property
We have $c_1 = m_{01} \times k^0 + m_{11} \times k^1 + \ldots m_{i1} \times k^i + r_1 \times p$ and $c_2 = m_{02} \times k^0 + m_{12} \times k^1 + \ldots m_{i2} \times k^j + r_2 \times p \Rightarrow Enc(m_1) + Enc(m_2) = m'_0 \times k^0 + m'_1 \times k^1 + \ldots m'_l \times k^l + r' \times p$ where $l = Max(i, j)$ and $m'_x \in \{0, .., k-1\}$ $\forall x \in \{0, .., l\}$, hence $Enc(m_1) + Enc(m_2) = Enc(m_1 + m_2)$

Homomorphic multiplicative property
$Enc(m_1) \times Enc(m_2) = m_{01} \times (m_{02} \times k^0 + m_{12} \times k^1 + \ldots m_{i2} \times k^j) + m_{11} \times (m_{02} \times k^0 + m_{12} \times k^1 + \ldots m_{i2} \times k^j) + \ldots m_{i1} \times (m_{02} \times k^0 + m_{12} \times k^1 + \ldots m_{i2} \times k^j) + r' \times p = m'_0 \times k^0 + m'_1 \times k^1 + \ldots m'_l \times k^l + r' \times p$ where $l = i \times j + i + j + 1$ and $m'_x \in \{0, .., k-1\}$ $\forall x \in \{0, .., l\}$, hence $Enc(m_1) \times Enc(m_2) = Enc(m_1 \times m_2)$

4.5 Security analysis

Cryptanalysis can be applied to many kinds of known attacks obtained from various running and encryption scenarios. The public keys are available for everyone in asymmetric encryption schemes which makes them more vulnerable than symmetric cryptosystems in particular when the offered public keys refer to some critical information. At this point, we will discuss and analyze the robustness and limitations of MNF-G. The adversary's goal is either to recover the secret information or to extract some confidential encryption primitives.

In (44), we have the two parts c_+ and $c_\times$. We start with c_+ and with the straightforward attacks that are possible in many areas. Thus, rendering the factorization of n is possible using several scenarios. The analysis in MNF-G is focused on the

factorization problem. From the pair for the MNF-G public-key, some theoretical mathematical proprieties can be exploited by various cryptanalytic attacks to reveal the private key if the trapdoor is not well locked. Certain known attacks exploit weaknesses in the small prime number factorization.

The probabilistic homomorphic encryption schemes suffer from collision attacks. Collision attacks can be explained as performing certain operations over ciphered texts in order to recover the whole or part of the original plain text. Therefore, the system will be compromised when a low entropy plain text ρ is used. To this end, some conditions should be followed to guarantee the effectiveness of HE. Using a large number p will increase the entropy ρ, which is more suitable for preventing collision attacks.

Let m be a plain text whose encryption using $pk = (k + r \times p)$ is $c = m' + m'' \times pk$ and its encryption using $pk' = (k + r' \times p)$ is $c' = m' + m'' \times pk'$. Hence, $(c - c')$ is an encryption of 0 which equals $r'' \times p$. This case leads to revealing the value of p which is considered as a trapdoor. In MNF-G, the public key must be calculated in a unique way for hiding the 0 encryption issue.

Figure (4.3) shows the influence of the plaintext fragmentation on the time to find a solution in brute-force attack, where the adversary starts to test all possible values of the secret key sk from smallest to largest or vice versa. In an asymmetric encryption scheme over integers like RSA where $c = m^d$ and $m = c^e$, the public key d is known, the adversary tries to calculate $m = c^e$ for any which m chosen. To find the secret key e, the adversary needs to test all possible values in the interval $\{1, e\}$. The same thing for the encryption schemes like basic El-Gamal, Paillier and BGN (introduced by Boneh-Goh-Nissim (131)). Assuming that t is the time to perform an operation and x the size of the secret key. The function which expresses the required time for a successful brute-force attack is $f(x) = x \times t$.

In the proposed system, we focus on fragmentation of message m by which the adversary must test all the possibilities of the secret key for each possible fragmentation, i.e., for fragmentation 1, $c = 1 \times sk + (m - 1)$ the adversary should try all the values of sk. For fragmentation 2, $c = 2 \times sk + (m - 2)$ the adversary should try all the values of sk, etc. In this case, we have $\frac{m}{2} \times x$ tests. Assuming the size of m is x, we will have $f(x) = \frac{x}{2} \times x \times t$. Finally, we will obtain:

$$f(x) = \frac{x^2}{2} \;\; in\ MNF - G\ and \;\; f(x) = x \;\; in\ other\ schemes \tag{4.15}$$

The encryption manner of MNF-G is more like an $m \times k$ cryptosystem. These encryption models are clearly prone to certain attacks. Given a public key (n, pk), the size of pk will determine the number of possible combinations of $(k_i,\ r_i \times p_i)$ such that $k_i + r_i \times p_i = pk$. An attacker may start to recover $r \times p$ or k using all possible combinations. This case leads to the fragmentation and factorization

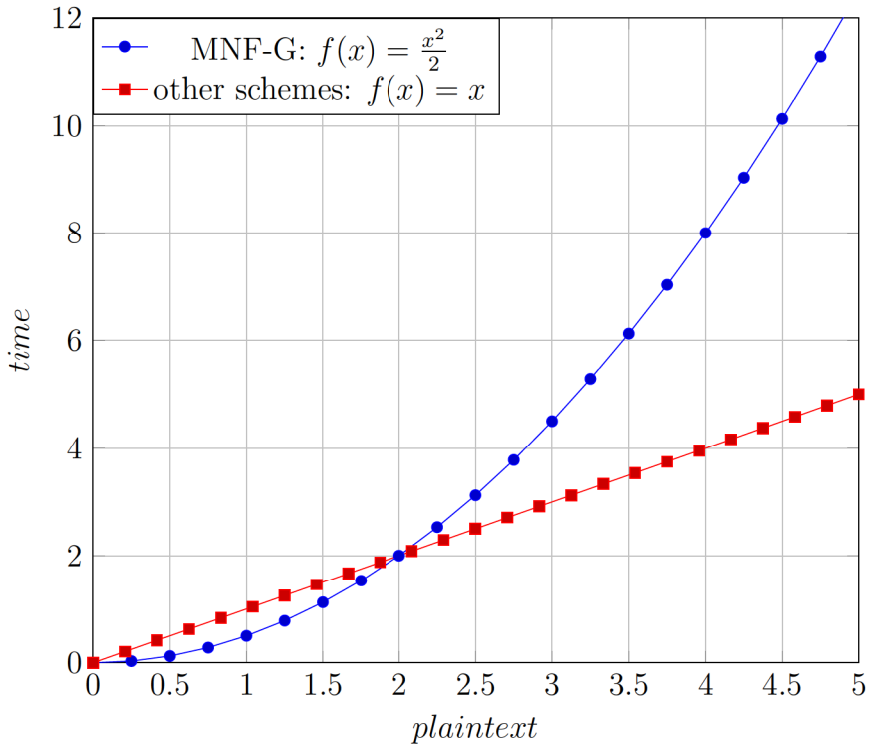


Figure 4.3: Time comparison for a successful brute-force attack

number problem. The attacker needs to make all possible combinations of k, and then factorize the remainder of $pk - k_i$, which amounts to trying to factorize $(pk-2)$ numbers. When pk is large, it is impossible to make all these factorizations with current technologies. The first part of MNF-G is a probabilistic scheme offering a sufficient condition to satisfy IND-CPA (Indistinguishability under Chosen Plaintext Attack). In reality, $c_\times$ is deterministic, which is not a weakness of the proposed scheme. In CPA, the attacker makes $c' = Enc(m')$ where m' is chosen, then he decrypts $c \times c'$ to get $m \times m'$ and he finds m. This type of attack is effective for some cryptosystem like RSA, because $c \times c' = (m \times m')^e$, i.e., $c \times c'$ only contains $m \times m'$ and the secret key. On the other hand, in MNF-G $c \times c' = m \times m' \times g^{m+m'}$, so it is more difficult to decipher it to obtain $m \times m'$.

The second part $c_\times$ is almost El-Gamal, where the power of the generator g is not random in each message. But it is the plaintext m itself, and that does not modify the scheme robustness which always depends on the discrete logarithm problem (DLP). The most attack on an El-Gamal cryptosystem is to solve the discrete logarithm problem: given both α and α^x from the finite field $\mathbb{Z}_p$, find the value for x. To make a hard DLP, it should select a prime p where $p-1$ has a large prime factor. However, if $p-1$ is only divisible by primes less than some positive integer β, the DLP can be reduced to finding a small number of size β instead of size $p-1$. The DLP can be solved for $log_\alpha(\beta)$ by computing $1, \alpha, \alpha^2, \ldots$

until finding a match with β, so that α can be replaced with $\alpha_1 = \alpha^m$ and β with $\beta_1 = \beta^m$ to solve $log_{\alpha_1}(\beta_1) \mod r$ such that $r \times m = p - 1$. Therefore, it should build the list $1, \alpha_1, \alpha_1^2, \ldots, \alpha_1^x$ of length at most r before finding β_1. El-Gamal cryptosystems are vulnerable to the man-in-the-middle attack which is a protocol for an eavesdropper where Eve is positioned between two communicators Alice and Bob to intercept their messages. According to the following scenario, Alice selects a secret key x, makes a public key (g, y_1) with $y_1 = g^{x_1}$, and sends it to Bob. This key is intercepted by Eve, who chooses a private random integer z, and makes another public key (g, y') where $y' = g^z$ which she sends to Bob, pretending to be Alice. Simultaneously, she sends that same key (g, y') to Alice, now pretending as Bob. The adversary Eve has set up a common session key $g^{x_1 \times z}$ with Alice and a common session key $g^{x_2 \times z}$ with Bob (noting that the public key of Bob is (g, y_2) where $y_2 = g^{x_2}$). Now, Eve can intercept the messages that are exchanged between Alice and Bob and can be decrypted, re-encrypted, and resent. To avoid this type of attack and that Bob would not have confused Eve with Alice, the public key of Alice could be confirmed by an independent certification authority. The classical El-Gamal in $\mathbb{Z}_p^*$ showed that it is secure against Chosen-Plaintext Attack (CPA) but not against Chosen-Ciphertext Attack (CCA) (132). In reality, for a scheme to be safe against CPA, the Decisional Diffie-Hellman assumption must be respected, which needs several settings, including secret key length, the p being a safe prime of the form $2 \times q + 1$ where q is also prime and the choice of a generator g which is a primitive root. Concerning CCA and MNF-G, a scheme is not secure against CCA if the encryption of a product is the product of the encryptions (e.g., $Enc(m_1 \times m_2) = Enc(m_1) \times Enc(m_2)$) like the RSA and El-Gamal, but in MNF-G $Enc(m_1 \times m_2) = (m_1 \times m_2) \times g^{m_1 \times m_2}$ and $Enc(m_1) \times Enc(m_2) = (m_1 \times m_2) \times g^{m_1 + m_2}$. So, the MNF-G is secure against CCA.

Complexity analysis: The proposed MNF-G scheme is presented into two parts. To encrypt the first part, we have three operations, fragmentation, multiplication, and addition. For the complexity reason of $C(n + 1) = C(n)$, the computational complexity of the first part equals $O(1)$. To encrypt the second part, we need to calculate g^m which is of complexity $O((log(n))^3)$. Contrariwise, if we use the proposed vec-power function (4.4.2), $C(n + 1) = C(n)$. In this case, the complexity becomes constant ($O(1)$). We have improved the efficiency and reduced the computational complexity compared with other schemes like (133) which is of complexity $O(n^4)$ or like (134) which is of complexity $O(n^13)$. The first part can be decrypted into two operations $((c \mod p) \mod (k - 1))$, that gives us a computational complexity equal to $O(1)$. To encrypt the second part, we need to calculate $(g^m)^{-1}$ which gives us a complexity equal to $O((log(n))^3)$. By using vec-power function, the complexity is equal to $O(1)$. Compared with (133) which is of complexity $O(n^4)$ in decryption or with (134) which is of complexity $O(n^12)$ in decryption, MNF-G offers a considerable computational complexity.

In DFE-PR, we rely on two types of robustness, the first is to recover the secret key p, so it is the factorization problem (on which the RSA is based) which consists in finding the decomposition into prime factors of an integer obtained secretly by multiplying two prime numbers of comparable size. For such a size of $n = p \times q$ there is until today no known method to find p, and mathematicians affirm that there is no algorithm running in polynomial time which gives the prime factors of any number. Factoring a number with 200 digits (670 bits) requires 4 billion years of machine computation when its complexity is sub-exponential $2^{o(n)}$, and the best algorithms for the factorization of integers have a complexity $O = 2^{n^{1/3}}$. The official journal of the General Secretariat of French National Defense [1] in the rules and recommendations concerning the choice and sizing of the mechanisms Standard robustness level cryptographic (Version 1.10), declares that the minimum size of n equals 2024 bits for use not to exceed the year 2020, and for use beyond 2020, the minimum size of n equals 4096 bits.

The second hardness side of our scheme is based on the Polynomial Reconstruction Problem to recover the private key k. The cryptosystems based on polynomial ring homomorphisms $R[x] \mapsto R[x]$ can be written in the form $c(x) = r(k(x))$ where $k(x)$ is a key and the most general formula $c(x) = r(x) \times k(x) + m$ like (135) scheme. In (136), the authors analyzed the security of this symmetric scheme against a known-plaintext attack (KPA), a ciphertext-only attack (COA), and a brute force attack (BFA), using two approaches. The first uses the univariate polynomial decomposition algorithm and runs in polynomial time, in numbered degrees and the size of n. The second method is based on the calculation of polynomial GCDs and polynomial factorization, and its complexity depends polynomially on the degree of the cyphertexts and logarithmically on the size of n. They discovered that it is very vulnerable to a known-plaintext attack. For COA, they found that the probability of finding the correct key is high enough, and the brute force attack finds a valid key with a probability $\rho \approx 1$. But all this with consideration that the diagram is symmetrical, i.e., without the noise of $r \times p$. This, of course, changes the balance, so that the addition of $r \times p$ modifies $k(x)$, so the three types of attack are ineffective with the two variables existing in the ciphertext, and so only the factorization remains with the attacker.

4.6 Implementation

To evaluate our proposed cryptosystem ((44)) and to prove that this cryptosystem is more efficient compared with other ones, we make three experiments, whose

[1] $http://circulaire.legifrance.gouv.fr/pdf/2009/04/cir_2304.pdf$

results are presented in Table 4.3. On the other hand, the result shown in Table 4.2 indicates another improvement that we obtained for our cryptosystem performance.

Table 4.3 presents the results of four schemes obtained by three tests for each one. MNF-G has been done under Python using an HP Laptop with the following specifications: Processor Intel(R) Core(TM) i3-3110M CPU @ 2.40 GHz, 2 Core(s), 4 Logical Processor(s), 4 Go RAM. TFHE-DDA (Targeted Fully Homomorphic Encryption Based on a Double Decryption Algorithm) has been related to the implementation of the scheme (127). The Yah-Scheme is related to (137) where the implementation is done using a personal computer with bi-cores Intel Core i5 CPU running at 2.40 GHz, 512KB L2 cache, and 4GB of RAM. The fourth line is devoted to the Than-Scheme (138) where the implementation of the proposed framework has been done in a private cloud using Eucalyptus running on a 2.50 GHz Intel Core i5-3120 M Processor and 16 GB RAM.

We have implemented the work of (127) with the aim to be able to make a comparison. Concerning the keygen time, the results were close due to the reconciliation of the variables, i.e., size and number. The reason behind the difference is that their scheme has an additional computation raised by $\beta = \alpha^p$. In fact, there are three operations, *Enc*, *Dec*, and *SumPrd*, which perform the addition and multiplication at the same time. As to *Enc*, the obtained results say that our HE time is better than the HE computation time of (127) for all tests. This is logical because we only have seven binary operations (4 in the first part + 3 in the second) between them and there is only one power operation. On the other hand, for the encryption of (127), which is made up of three parts, they have nine binary operations (3 + 3 + 3) with three power operations.

Decryption is also an essential operation often done by the client, who quite often has only low computational resources. The total time of the MNF-G decryption (c_+ and $c_\times$) is almost equal to the decryption time of the second part of (127). The main difference between MNF-G and (127) lies in the decryption of the third part ($c_3 = \beta^{m_1+m_2}$) of (127) which needs the use of discrete logarithm to recover the addition result. We have implemented three algorithms to decrypt β^m which are Pohling Hellman, Pollard rho, and Baby-step giant step, but none of them could get the result within a reasonable time compared with MNF-G. In terms of size, MNF-G uses two values that represent ciphertext (of order $2 \times n$) and average pk size of $\frac{5}{2} \times n$, while (127) uses three values (of order $3 \times n$) and average pk size of $2 \times n$.

In (139), J.S.Coron et al. have used a single-core desktop computer with an Intel Core2 Duo E8500 CPU at 3.12 GHz. When they implemented their scheme using the secret key length $\eta = 2176$ bits which in MNF-G uses $p \approx 700$ digits ≈ 2300 bits, the consumed *Enc* processing time equals 10 s (vs. 13.7 ms in MNF-G) and the consumed *Dec* processing time equals 0.02 s (vs. 19.4 ms in MNF-G) with a

MNF-G test(1)	**result** (ms)	**MNF-G test(2)**	**result** (ms)	**MNF-G test(3)**	**result** (ms)
$p = 24$ digits	Keygen : 0.307	$p = 150$ digits	Keygen : 24.8	$p = 300$ digits	Keygen : 167
$k = 16$ digits	Enc : 0.053	$k = 100$ digits	Enc : 0.638	$k = 200$ digits	Enc : 3.92
$g = 8$ digits	Dec : 0.118	$g = 20$ digits	Dec : 1.20	$g = 40$ digits	Dec : 5.83
$m = 8$ digits	SumMult : 0.009	$m = 50$ digits	SumMult : 0.017	$m = 100$ digits	SumMult : 0.037
TFHE-DDA test(1)	**result** (ms)	**TFHE-DDA test(2)**	**result** (ms)	**TFHE-DDA test(3)**	**result** (ms)
$p = 24$ digits	Keygen : 0.396	$p = 150$ digits	Keygen : 29.1	$p = 300$ digits	Keygen : 195
$k = 16$ digits	Enc : 0.109	$k = 100$ digits	Enc : 3.07	$k = 200$ digits	Enc : 17.1
$g = 8$ digits	Dec : » 1hr	$g = 20$ digits	Dec : » 1hr	$g = 40$ digits	Dec : » 1hr
$m = 8$ digits	SumMult : 0.011	$m = 50$ digits	SumMult : 0.033	$m = 100$ digits	SumMult : 0.088
Yah-Scheme test(1)	**result** (ms)	**Yah-Scheme test(2)**	**result** (ms)	**Yah-Scheme test(3)**	**result** (ms)
$sk = 280$ digits	Keygen : 120	$sk = 560$ digits	Keygen : 310	$sk = 1120$ digits	Keygen : 1200
$m = 140$ digits	Enc : 20	$m = 280$ digits	Enc : 40	$m = 560$ digits	Enc : 190
/	Dec : 3	/	Dec : 4	/	Dec : 10
/	Sum : « 1	/	Sum : « 1	/	Sum : « 1
/	Mult : « 1	/	Mult : 1	/	Mult : 2
Than-Scheme test(1)	**result** (ms)	**Than-Scheme test(2)**	**result** (ms)	**Than-Scheme test(3)**	**result** (ms)
$m = 8$ digits	Enc : 47	$m = 64$ digits	Enc : 47	$m = 128$ digits	Enc : 62
/	Dec : 15	/	Dec : 15	/	Dec : 21

Table 4.3: Processing time comparison

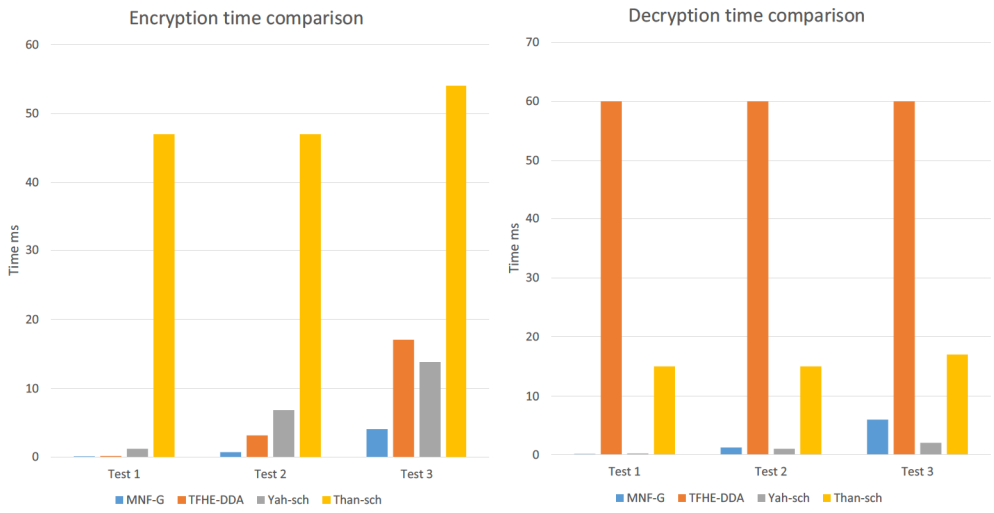


Figure 4.4: Encryption Decryption time comparison

pk size of 89 MB (vs. 7.5 KB in MNF-G). Table 4.3 shows the efficiency of MNF-G compared with other schemes. This efficiency has been achieved by reducing complexity and avoiding the use of additional operations such as bootstrapping, relinearization, flattening, etc. Figure 4.4 represents the experimental results of schemes MNF-G, TFHE-DDA, Yah-scheme, and Than-scheme where time=60 expresses time >> 1 hour. In (45), we focused on studying the results of the number of addition and multiplication operations for this technique, but we mentioned a simple overview of the results related to the speed of execution. Table 4.4 shows these results which have been defined under Python using a HP Laptop with the following specifications: Processor Intel(R) Core(TM) i3-3110M CPU @ 2.40 GHz, 2 Core(s), 4 Logical Processor(s), 4 Go RAM. Using a private key with 128 bits, a secret key with 1160 bits, and a message with 40 bits. The authors of (140)

Scheme	Enc	Dec
ours	0.00014 s	0.0239 s
(140)	0.899 s	0.785 s
(139)	0.05 s	0.01 s
(141)	0.255 s	0.493 s

Table 4.4: Processing time

performed their experiments on an Intel Xeon W3565 CPU for scheme (142) of a private key with 128 bits, where in (139) a single-core desktop computer with an Intel Core2 Duo E8500 CPU at 3.12 GHz with a secret key length $\eta = 1088$ bits and toy parameters have been used, as in (141). The proposed scheme was

implemented in C using the GNU multi-precision library (GMP)on a 3 GHz processor with 4 GB RAM with the polynomial length $d(x) = 128$ bits and 5 bits of message length.

For an encryption operation, we have $c = m_0 \times k^0 + m_1 \times k^1 + \ldots m_{d-1} \times k^{d-1}$, according to Equation 2, $c < 10 \times k^{d-1}$ with $k > 9 \Rightarrow p > 10 \times 9^{d-1}$ where d is the number of digits of m, with $t-1$ addition operations, $C = \sum_{i=1}^{t} c_i \Rightarrow k > (9+9+\cdots+9) \Rightarrow k > 9 \times t$, $p > \sum_{i=1}^{t} 10 \times k^{d-1} \Rightarrow p > 10 \times t \times (9 \times t)^{d-1}$. Concerning multiplication, $C = \prod_{i=1}^{t} c_i \Rightarrow k > \sum_{i=1}^{t-1}(d \times (\prod_{j=1}^{t} m_j))$, with $Max(m_i) = 9 \Rightarrow k > (t-1) \times d \times 9^t$, $p > \prod_{i=1}^{t} 10 \times k^{d-1} \Rightarrow p > 10^t \times ((t-1) \times d \times 9^t)^{t \times (d-1)}$ where $d = length(m)$ and $(t-1)$ = number of operations. The Table 4.5 (length in digit) shows some results for practical examples:

m	k	p	# of operations
2	24	566	23
5	14	646	12
10	10	691	8

Table 4.5: Number of multiplication operations

4.7 Conclusion

In the literature, the nature of cloud security has led researchers to cope with the homomorphic issue. Many proposed homomorphic schemes emit a highly randomized noise during homomorphic operations, due to uncontrolled noise generation. In fact, few proposed schemes are practical to be executed on ciphertexts. Considering cloud security challenges, we have presented in this chapter lightweight and efficient encryption schemes.

MNF-G and DFE-PR introduce a new term into asymmetric cryptography where fragmentation offers linear encryption for allowing to perform a practical homomorphy over integers, this encryption is adapted to the characteristics of cloud computing. We showed the cryptanalyses and saw the results of the implementation of the proposed schemes which show their feasibility and their efficiency compared to other schemes.

Chapter 5

General Conclusion

In this work, we have addressed the problematic of the security challenges imposed by the widespread use of Internet of Things networks in our daily lives. The rapid growth in communication technology, embedded systems, and interconnectivity among smart devices has made it possible to build Internet of Things networks as we know them today. By relying on a group of sensors and small smart devices at low cost, it has become possible to change the environment around us into a smart environment known today as smart cities and smart homes.The expansion of the communication area of IoT devices and the collection and processing of a huge amount of exchanged information, with the restricted storage and processing capabilities of an IoT object, have led to the emergence of many problems in terms of security, privacy, and confidentiality of information transmission.

To address these problems, we first started by giving an overview of distributed systems and cloud computing, their architecture and utility, and how the architecture of the Internet of Things is related to a distributed architecture. We also discussed the security concerns in these distributed environments and how to enforce security measures to keep this environment secure.

The second chapter discussed the Internet of Things environment and how to secure it, where we began by providing a comprehensive view of the Internet of Things architecture and its components due to the growth of technology in the field of communications and the interconnectivity of smart devices. This section shows that the limited capabilities in storage and processing of an IoT object by the available resources are often very constrained due to size, energy, power, and computational capability limitations. We detailed that the primary research challenge is to ensure that we get the correct data at the expected level of accuracy. We explain that along with data collection and processing, there are also challenges in connectivity and data communication. We have presented the encryption techniques and approaches for data protection and how they can be adopted to secure the IoT environment.

Our contribution is the subject of the third chapter. we have proposed the FlexenTech encryption technique to cope with the device limitation challenge to render these devices more applicable in terms of security requirements. The mechanism of the proposed technique reduces the computation time incurred by data transmission, because we have minimized the number of rounds used to cipher the information and introduced many improvements to minimize computation

The fourth chapter presented our collaborative work contributions to encryption. They emphasize the problem of the confidentiality of data stored in the Cloud. We have proposed two homomorphic encryption techniques. The objective is to make cloud computing entities in an IoT environment independent by allowing them to perform different operations on encrypted data. Because these methods are based on linear and modular expressions, they are easy to use. We have proven their homomorphic characteristics mathematically. The security analysis of these schemes shows that they are feasible and extremely efficient in execution time compared to other schemes.

In conclusion, we can say that to address the problems posed in the field of information security in general and in the field of distributed systems in particular, we presented a study of the various technologies and concepts in this field. In this thesis, we have offered solutions to manage and control data traffic, secure the Internet of Things environment, and achieve privacy for network users. We have attempted to adopt current distributed systems techniques and cloud computing solutions to apply them to the Internet of Things environment while considering device limitations, resource limitations, and computational power limitations. The results provided were promising, and they can be used to build new techniques that increase the reliability of these components and ensure the security and integrity of IoT networks.

Bibliography

[1] Simone Cirani, Gianluigi Ferrari, Marco Picone, and Luca Veltri. *Internet of things: architectures, protocols and standards*. John Wiley & Sons, 2018.

[2] Pallavi Sethi and Smruti R Sarangi. Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017, 2017.

[3] Maarten van Steen and Andrew S Tanenbaum. A brief introduction to distributed systems. *Computing*, 98(10):967–1009, 2016.

[4] Mohammad Yasser Chuttur and Jay Mungur. *Cloud Computing Fundamentals*. Le Printemps Ltee, 2021.

[5] Michael Fisher, Viviana Mascardi, Kristin Yvonne Rozier, Bernd-Holger Schlingloff, Michael Winikoff, and Neil Yorke-Smith. Towards a framework for certification of reliable autonomous systems. *Autonomous Agents and Multi-Agent Systems*, 35(1):1–65, 2021.

[6] Sadatoshi Kumagai and Toshiyuki Miyamoto. Autonomous distributed system and its realization by multi agent nets. In *International Conference on Application and Theory of Petri Nets*, pages 3–19. Springer, 2007.

[7] Abhijit Belapurkar, Anirban Chakrabarti, Harigopal Ponnapalli, Niranjan Varadarajan, Srinivas Padmanabhuni, and Srikanth Sundarrajan. *Distributed systems security: issues, processes and solutions*. John Wiley & Sons, 2009.

[8] Bijeta Seth, Surjeet Dalal, Vivek Jaglan, Dac-Nhuong Le, Senthilkumar Mohan, and Gautam Srivastava. Integrating encryption techniques for secure data storage in the cloud. *Transactions on Emerging Telecommunications Technologies*, 33(4):e4108, 2022.

[9] Sajid Habib Gill, Mirza Abdur Razzaq, Muneer Ahmad, Fahad M Almansour, Ikram Ul Haq, NZ Jhanjhi, Malik Zaib Alam, and Mehedi Masud. Security and privacy aspects of cloud computing: a smart campus case study. *Intelligent Automation And Soft Computing*, 31(1):117–128, 2022.

[10] Aysha Albarqi, Ethar Alzaid, Fatimah Al Ghamdi, Somaya Asiri, Jayaprakash Kar, et al. Public key infrastructure: A survey. *Journal of Information Security*, 6(01):31, 2014.

[11] Vincent Lozupone. Analyze encryption and public key infrastructure (pki). *International Journal of Information Management*, 38(1):42–44, 2018.

[12] Fred Piper and Sean Murphy. *Cryptography: A very short introduction*, volume 68. Oxford Paperbacks, 2002.

[13] Mourade Azrour, Jamal Mabrouki, Azidine Guezzaz, and Ambrina Kanwal. Internet of things security: challenges and key issues. *Security and Communication Networks*, 2021:1–11, 2021.

[14] Crystal Panek. *Understanding authentication, authorization, and accounting*, pages 33–109. 2020.

[15] James Manyika and Michael Chui. By 2025, internet of things applications could have $11 trillion impact. *Insight Publications*, 2015.

[16] Dorsaf Swessi and Hanen Idoudi. A survey on internet-of-things security: threats and emerging countermeasures. *Wireless Personal Communications*, 124(2):1557–1592, 2022.

[17] Pintu Kumar Sadhu, Venkata P Yanambaka, and Ahmed Abdelgawad. Internet of things: Security and solutions survey. *Sensors*, 22(19):7433, 2022.

[18] Sanjeev Kaushik Ramani and SS Iyengar. Evolution of sensors leading to smart objects and security issues in iot. In *International Symposium on Sensor Networks, Systems and Security*, pages 125–136. Springer, 2017.

[19] Sachin Kumar, Prayag Tiwari, and Mikhail Zymbler. Internet of things is a revolutionary approach for future technology enhancement: a review. *Journal of Big data*, 6(1):1–21, 2019.

[20] Pawel Sniatala, SS Iyengar, and Sanjeev Kaushik Ramani. Smart objects in cyber-physical systems. In *Evolution of Smart Sensing Ecosystems with Tamper Evident Security*, pages 9–16. Springer, 2021.

[21] R Samani. key security challenges for the internet of things, 2017.

[22] Pawel Sniatala, S.S. Iyengar, and Sanjeev Kaushik Ramani. *IoT Security*, pages 17–24. Springer International Publishing, Cham, 2021.

[23] Fadi Al-Turjman, Hadi Zahmatkesh, and Ramiz Shahroze. An overview of security and privacy in smart cities' iot communications. *Transactions on Emerging Telecommunications Technologies*, 33(3):e3677, 2022.

[24] Sarhad Arisdakessian, Omar Abdel Wahab, Azzam Mourad, Hadi Otrok, and Mohsen Guizani. A survey on iot intrusion detection: Federated learning, game theory, social psychology and explainable ai as future directions. *IEEE Internet of Things Journal*, 2022.

[25] Lo'ai Tawalbeh, Fadi Muheidat, Mais Tawalbeh, and Muhannad Quwaider. Iot privacy and security: Challenges and solutions. *Applied Sciences*, 10(12), 2020.

[26] P.P. Ray. A survey on internet of things architectures. *Journal of King Saud University - Computer and Information Sciences*, 30(3):291–319, 2018.

[27] Ammar Rayes and Samer Salam. *IoT Protocol Stack: A Layered View*, pages 93–138. Springer International Publishing, Cham, 2017.

[28] Gustavus J Simmons. Cryptography.

[29] Fred Cohen. A short history of cryptography. *http://www. all. net/books/ip/Chap2-1. html*, 1990.

[30] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM review*, 45(4):727–784, 2003.

[31] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[32] Donald Nokam Kuaté. *Cryptographie homomorphe et transcodage d'image/video dans le domaine chiffré*. PhD thesis, Université Paris Saclay (COmUE), 2018.

[33] V Biksham and D Vasumathi. A lightweight fully homomorphic encryption scheme for cloud security. *International Journal of Information and Computer Security*, 13(3-4):357–371, 2020.

[34] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 24–43, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[35] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *International Conference on Smart Card Research and Advanced Applications*, pages 277–284. Springer, 1998.

[36] Patrik Ekdahl and Thomas Johansson. A new version of the stream cipher snow. In *International Workshop on Selected Areas in Cryptography*, pages 47–61. Springer, 2002.

[37] Morris Dworkin, Elaine Barker, James Nechvatal, James Foti, Lawrence Bassham, E. Roback, and James Dray. Advanced encryption standard (aes), 2001-11-26 2001.

[38] Tariq Jamil. The rijndael algorithm. *IEEE potentials*, 23(2):36–38, 2004.

[39] N Penchalaiah and R Seshadri. Effective comparison and evaluation of des and rijndael algorithm (aes). *International journal of computer science and engineering*, 2(05):1641–1645, 2010.

[40] Bibhudendra Acharya, Saroj Kumar Panigrahy, Sarat Kumar Patra, and Ganapati Panda. Image encryption using advanced hill cipher algorithm. *International Journal of Recent Trends in Engineering*, 1(1):663–667, 2009.

[41] Ziad E Dawahdeh, Shahrul N Yaakob, and Rozmie Razif bin Othman. A new image encryption technique combining elliptic curve cryptosystem with hill cipher. *Journal of King Saud University-Computer and Information Sciences*, 30(3):349–355, 2018.

[42] Stéphane Jacob. *Protection cryptographique des bases de données: conception et cryptanalyse*. PhD thesis, 2012.

[43] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[44] Mostefa Kara, Abdelkader Laouid, Mohammed Amine Yagoub, Reinhardt Euler, Saci Medileh, Mohammad Hammoudeh, Amna Eleyan, and Ahcène Bounceur. A fully homomorphic encryption based on magic number fragmentation and el-gamal encryption: Smart healthcare use case. *Expert Systems*, pages 1–10, 2021.

[45] Mostefa Kara, Abdelkader Laouid, Reinhardt Euler, Mohammed Amine Yagoub, Ahcene Bounceur, Mohammad Hammoudeh, and Saci Medileh. A homomorphic digit fragmentation encryption scheme based on the polynomial reconstruction problem. In *The 4th International Conference on Future Networks and Distributed Systems (ICFNDS)*, pages 1–6, 2020.

[46] Duong Hieu Phan. *Sécurité et efficacité des schémas cryptographiques*. PhD thesis, Ecole Polytechnique X, 2005.

[47] Pawel Sniatala, S.S. Iyengar, and Sanjeev Kaushik Ramani. *Cryptosystems: Foundations and the Current State-of-the-Art*, pages 47–53. Springer International Publishing, Cham, 2021.

[48] Tech Guides. Fasttrack To Cryptography uses of cryptography, 2022.

[49] Maulin Patel and Jianfeng Wang. Applications, challenges, and prospective in emerging body area networking technologies. *IEEE Wireless communications*, 17(1), 2010.

[50] Blaise Laurent Patrick Gassend. *Physical random functions*. PhD thesis, Massachusetts Institute of Technology, 2003.

[51] Ulrich Rührmair, Frank Sehnke, Jan Sölter, Gideon Dror, Srinivas Devadas, and Jürgen Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM Conference on Computer and Communication Security*, pages 237–249. ACM, 2010.

[52] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 22–41. Springer, 2014.

[53] Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.

[54] Bart Preneel and Tsuyoshi Takagi. *Cryptographic Hardware and Embedded Systems–CHES 2011: 13th International Workshop, Nara, Japan, September 28–October 1, 2011, Proceedings*, volume 6917. Springer, 2011.

[55] Peter Gutmann, David Naccache, and Charles C Palmer. When hashes collide [applied cryptography]. *IEEE security & privacy*, 3(3):68–71, 2005.

[56] Jens-Peter Kaps, Gunnar Gaubatz, and Berk Sunar. Cryptography on a speck of dust. *Computer*, 40(2), 2007.

[57] Sana Belguith, Nesrine Kaaniche, Mohammad Hammoudeh, and Tooska Dargahi. Proud: verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted iot applications. *Future Generation Computer Systems*, 2019.

[58] James Carey. Protection of personal data contained on an rfid-enabled device, September 11 2018. US Patent App. 15/445,328.

[59] Nouha Oualha and Kim Thuat Nguyen. Lightweight attribute-based encryption for the internet of things. In *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*, pages 1–6. IEEE, 2016.

[60] Xuanxia Yao, Zhi Chen, and Ye Tian. A lightweight attribute-based encryption scheme for the internet of things. *Future Generation Computer Systems*, 49:104–112, 2015.

[61] Muath AlShaikh, Lamri Laouamer, Laurent Nana, and Anca Chrisitine Pascu. Efficient and robust encryption and watermarking technique based on a new chaotic map approach. *Multimedia Tools and Applications*, 76(6):8937–8950, 2017.

[62] Hanguang Luo, Guangjun Wen, Jian Su, and Zhong Huang. Slap: Succinct and lightweight authentication protocol for low-cost rfid system. *Wireless Networks*, 24(1):69–78, 2018.

[63] Zilong Liu, Dongsheng Liu, Lun Li, Hui Lin, and Zhenqiang Yong. Implementation of a new rfid authentication protocol for epc gen2 standard. *IEEE Sensors Journal*, 15(2):1003–1011, 2015.

[64] Mark A Will and Ryan KL Ko. A guide to homomorphic encryption. *The Cloud Security Ecosystem: Technical, Legal, Business and Management Issues*, pages 101–127, 2015.

[65] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[66] T. Elgamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[67] David Naccache and Jacques Stern. A new public key cryptosystem based on higher residues. In *Proceedings of the 5th ACM Conference on Computer and Communications Security*, CCS '98, page 59–66, New York, NY, USA, 1998. Association for Computing Machinery.

[68] Steven D Galbraith. Elliptic curve paillier schemes. *Journal of Cryptology*, 15(2):129–138, 2002.

[69] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In *International Workshop on Public Key Cryptography*, pages 315–329. Springer, 2007.

[70] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ACM Transactions on Computation Theory (TOCT)*, 6(3):1–36, 2014.

[71] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography*, pages 325–341, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[72] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In *Theory of Cryptography Conference*, pages 575–594. Springer, 2007.

[73] Zvika Brakerski. Fundamentals of fully homomorphic encryption-a survey. In *Electron. Colloquium Comput. Complex.*, volume 25, page 125, 2018.

[74] Pawel Sniatala, SS Iyengar, and Sanjeev Kaushik Ramani. Homomorphic encryption. In *Evolution of Smart Sensing Ecosystems with Tamper Evident Security*, pages 69–76. Springer, 2021.

[75] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *International Workshop on Public Key Cryptography*, pages 420–443. Springer, 2010.

[76] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 169–178, 2009.

[77] Frederik Armknecht, Colin Boyd, Christopher Carr, Kristian Gjøsteen, Angela Jäschke, Christian A Reuter, and Martin Strand. A guide to fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2015:1192, 2015.

[78] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (CSUR)*, 51(4):1–35, 2018.

[79] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 24–43. Springer, 2010.

[80] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234, 2012.

[81] Saci Medileh, Abdelkader Laouid, Reinhardt Euler, Ahcène Bounceur, Mohammad Hammoudeh, Muath AlShaikh, Amna Eleyan, Osama Ahmed Khashan, et al. A flexible encryption technique for the internet of things environment. *Ad Hoc Networks*, 106:102240, 2020.

[82] Vikas Hassija, Vinay Chamola, Vikas Saxena, Divyansh Jain, Pranav Goyal, and Biplab Sikdar. A survey on iot security: application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743, 2019.

[83] Maria Gulzar and Ghulam Abbas. Internet of things security: A survey and taxonomy. In *2019 International Conference on Engineering and Emerging Technologies (ICEET)*, pages 1–6. IEEE, 2019.

[84] Jaber Hosseinzadeh and Maghsoud Hosseinzadeh. A comprehensive survey on evaluation of lightweight symmetric ciphers: hardware and software implementation. *Advances in Computer Science: an International Journal*, 5(4):31–41, 2016.

[85] Daniel Engels, Markku-Juhani O Saarinen, Peter Schweitzer, and Eric M Smith. The hummingbird-2 lightweight authenticated encryption algorithm.

In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, pages 19–31. Springer, 2011.

[86] Wenling Wu and Lei Zhang. Lblock: a lightweight block cipher. In *International Conference on Applied Cryptography and Network Security*, pages 327–344. Springer, 2011.

[87] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef. Internet of things security: A top-down survey. *Computer Networks*, 2018.

[88] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. Simon and speck: Block ciphers for the internet of things. *IACR Cryptology ePrint Archive*, 2015:585, 2015.

[89] Manish Kumar, Sunil Kumar, Rajat Budhiraja, MK Das, and Sanjeev Singh. Lightweight data security model for iot applications: A dynamic key approach. In *Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2016 IEEE International Conference on*, pages 424–428. IEEE, 2016.

[90] Muhammad Usman, Irfan Ahmed, M Imran Aslam, Shujaat Khan, and Usman Ali Shah. Sit: a lightweight encryption algorithm for secure internet of things. *arXiv preprint arXiv:1704.08688*, 2017.

[91] Saurabh Singh, Pradip Kumar Sharma, Seo Yeon Moon, and Jong Hyuk Park. Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2017.

[92] Rolando Montero-Canela, Ernesto Zambrano-Serrano, Edna I Tamariz-Flores, Jesus M Muñoz-Pacheco, and Richard Torrealba-Meléndez. Fractional chaos based-cryptosystem for generating encryption keys in ad hoc networks. *Ad Hoc Networks*, 97:102005, 2020.

[93] Sana Belguith, Nesrine Kaaniche, and Giovanni Russello. Pu-abe: Lightweight attribute-based encryption supporting access policy update for cloud assisted iot. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 924–927. IEEE, 2018.

[94] Steve Moffat, Mohammad Hammoudeh, and Robert Hegarty. A survey on ciphertext-policy attribute-based encryption (cp-abe) approaches to data security on mobile devices and its application to iot. In *Proceedings of the International Conference on Future Networks and Distributed Systems*, page 34. ACM, 2017.

[95] Sanaah Al Salami, Joonsang Baek, Khaled Salah, and Ernesto Damiani. Lightweight encryption for smart home. In *2016 11th International Conference on Availability, Reliability and Security (ARES)*, pages 382–388. IEEE, 2016.

[96] Jiguo Li, Yichen Zhang, Jianting Ning, Xinyi Huang, Geong Sen Poh, and Debang Wang. Attribute based encryption with privacy protection and accountability for cloudiot. *IEEE Transactions on Cloud Computing*, 2020.

[97] Sana Belguith, Nesrine Kaaniche, Giovanni Russello, et al. Lightweight attribute-based encryption supporting access policy update for cloud assisted iot. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications-Volume 1: SECRYPT*, pages 135–146. SciTePress, 2018.

[98] Olivier Alphand, Michele Amoretti, Timothy Claeys, Simone Dall'Asta, Andrzej Duda, Gianluigi Ferrari, Franck Rousseau, Bernard Tourancheau, Luca Veltri, and Francesco Zanichelli. Iotchain: A blockchain security architecture for the internet of things. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2018.

[99] Ludwig Seitz, Goeran Selander, Erik Wahlstroem, Samuel Erdtman, and Hannes Tschofenig. Authentication and authorization for constrained environments (ace). *Internet Engineering Task Force*, 2017.

[100] Mališa Vučinić, Bernard Tourancheau, Franck Rousseau, Andrzej Duda, Laurent Damon, and Roberto Guizzetti. Oscar: Object security architecture for the internet of things. *Ad Hoc Networks*, 32:3–16, 2015.

[101] Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Léo Perrin, Johann Großschädl, and Alex Biryukov. Triathlon of lightweight block ciphers for the internet of things. *Journal of Cryptographic Engineering*, pages 1–20, 2015.

[102] Sreeja Rajesh, Varghese Paul, Varun G Menon, and Mohammad R Khosravi. A secure and efficient lightweight symmetric encryption scheme for transfer of text files between embedded iot devices. *Symmetry*, 11(2):293, 2019.

[103] Abdelkader Laouid, Abdelnasser Dahmani, Hani Ragab Hassen, Ahcène Bounceur, Reinhardt Euler, Farid Lalem, and Abdelkamel Tari. A self-managing volatile key scheme for wireless sensor networks. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–16, 2018.

[104] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge university press, 2009.

[105] Abdelkader Laouid, Mohamed-Lamine Messai, Ahcène Bounceur, Reinhardt Euler, Abdelnasser Dahmani, and Abdelkamel Tari. A dynamic and distributed key management scheme for wireless sensor networks. In *Proceedings of the International Conference on Internet of things and Cloud Computing*, page 70. ACM, 2016.

[106] Chi-Yuan Chen and Han-Chieh Chao. A survey of key distribution in wireless sensor networks. *Security and Communication Networks*, 7(12):2495–2508, 2014.

[107] Seung-Hyun Seo, Jongho Won, Salmin Sultana, and Elisa Bertino. Effective key management in dynamic wireless sensor networks. *IEEE Transactions on Information Forensics and Security*, 10(2):371–383, 2015.

[108] Rafael Alvarez, Cándido Caballero-Gil, Juan Santonja, and Antonio Zamora. Algorithms for lightweight key exchange. *Sensors*, 17(7):1517, 2017.

[109] Craig Costello and Patrick Longa. Four$\mathbb{Q}$: Four-dimensional decompositions on a $\mathbb{Q}$-curve over the mersenne prime. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 214–235. Springer, 2015.

[110] Seokhie Hong, Deukjo Hong, Youngdai Ko, Donghoon Chang, Wonil Lee, and Sangjin Lee. Differential cryptanalysis of tea and xtea. In *International Conference on Information Security and Cryptology*, pages 402–417. Springer, 2003.

[111] Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 30–35, 2003.

[112] Norah Alassaf, Adnan Gutub, Shabir A Parah, and Manal Al Ghamdi. Enhancing speed of simon: a light-weight-cryptographic algorithm for iot applications. *Multimedia Tools and Applications*, 78(23):32633–32657, 2019.

[113] Tingyuan Nie, Chuanwang Song, and Xulong Zhi. Performance evaluation of des and blowfish algorithms. In *2010 International Conference on Biomedical Engineering and Computer Science*, pages 1–4. IEEE, 2010.

[114] Phongsak Prasithsangaree and Prashant Krishnamurthy. Analysis of energy consumption of rc4 and aes algorithms in wireless lans. In *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*, volume 3, pages 1445–1449. IEEE, 2003.

[115] David J Wheeler and Roger M Needham. Tea, a tiny encryption algorithm. In *International Workshop on Fast Software Encryption*, pages 363–366. Springer, 1994.

[116] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[117] Davender S Malik, John M Mordeson, and MK Sen. *Fundamentals of abstract algebra*. McGraw-Hill, 1997.

[118] Xun Yi, Russell Paulet, and Elisa Bertino. Homomorphic encryption. In *Homomorphic encryption and applications*, pages 27–46. Springer, 2014.

[119] Shafi Goldwasser and Silvio Micali. Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC '82, page 365–377, New York, NY, USA, 1982. Association for Computing Machinery.

[120] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Theory of Cryptography*, pages 325 341, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[121] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *Annual International Cryptology Conference*, pages 565–596. Springer, 2018.

[122] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

[123] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 129–148. Springer, 2011.

[124] Mohammed Amine Yagoub, Laouid Abdelkader, Okba Kazar, Ahcène Bounceur, Reinhardt Euler, and Muath AlShaikh. An adaptive and efficient fully homomorphic encryption technique. In *ICFNDS18*, pages 1–6, 06 2018.

[125] James Dyer, Martin Dyer, and Jie Xu. Practical homomorphic encryption over the integers for secure computation in the cloud. *International Journal of Information Security*, 18(5):549–579, 2019.

[126] Yarkın Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. Toward practical homomorphic evaluation of block ciphers using prince. In *International Conference on Financial Cryptography and Data Security*, volume 8438, pages 208–220, 03 2014.

[127] Yatao Yang, Shuang Zhang, Junming Yang, Jia Li, and Zichen Li. Targeted fully homomorphic encryption based on a double decryption algorithm for polynomials. *Tsinghua science and technology*, 19(5):478–485, 2014.

[128] Fatty M Salem and Ruhul Amin. A privacy-preserving rfid authentication protocol based on el-gamal cryptosystem for secure tmis. *Information Sciences*, 527:382–393, 2020.

[129] Fang-Yu Rao. On the security of a variant of elgamal encryption scheme. *IEEE Transactions on Dependable and Secure Computing*, 2017.

[130] Anees Ara, Mznah Al-Rodhaan, Yuan Tian, and Abdullah Al-Dhelaan. A secure privacy-preserving data aggregation scheme based on bilinear elgamal cryptosystem for remote health monitoring systems. *IEEE Access*, 5:12601–12617, 2017.

[131] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of cryptography conference*, pages 325–341. Springer, 2005.

[132] Marc Joye. Secure elgamal-type cryptosystems without message encoding. In *The New Codebreakers*, pages 470–478. Springer, 2016.

[133] Keke Gai, Meikang Qiu, Yujun Li, and Xiao-Yang Liu. Advanced fully homomorphic encryption scheme over real numbers. In *2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud)*, pages 64–69. IEEE, 2017.

[134] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 315–335. Springer, 2013.

[135] AO Zhirov, OV Zhirova, and SF Krendelev. Bezopasnye oblachnye vychisleniya s pomoshhyu homomorfnoj cryptographii. *BIT (bezopasnost'informacionnyx technology) journal*, 1:6–12, 2013.

[136] Alina Trepacheva. Cryptanalysis of polynomial based homomorphic encryption. In *ACM International Conference Proceeding Series*, volume 2014, pages 205–210, 09 2014.

[137] Ahmed El-Yahyaoui and Mohamed Dafir Ech-Cherif El Kettani. An efficient fully homomorphic encryption scheme. *IJ Network Security*, 21(1):91–99, 2019.

[138] M Thangavel and P Varalakshmi. Enhanced dna and elgamal cryptosystem for secure data storage and retrieval in cloud. *Cluster Computing*, 21(2):1411–1437, 2018.

[139] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, pages 487–504, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[140] Derian Boer and Stefan Kramer. Secure sum outperforms homomorphic encryption in (current) collaborative deep learning. *arXiv preprint arXiv:2006.02894*, 2020.

[141] Smaranika Dasgupta and S.K. Pal. Design of a polynomial ring based symmetric homomorphic encryption scheme. *Perspectives in Science*, 8, 07 2016.

[142] Le Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security*, PP:1–1, 12 2017.