Serial No:

People's Democratic Republic of Algeria Ministry Of Higher Education and Scientific Research

ECHAHID HAMMA LAKHDAR UNIVERSITY - EL OUED

FACULTY OF EXACT SCIENCES

COMPUTER SCIENCE DEPARTMENT

Thesis of Doctorate

Field: Mathematics and Computer Science
Sector: IT
Specialty: Advanced Information Systems

Title

The secure management of autonomous cloud entities in a distributed system

By: Mostefa Kara

Presented on: 22/ 12/ 2022

Composition of the jury:

Mr. Mohamed-khireddine Kholladi, Professor, University of El Oued, Chair

Mr. Abdelkader Laouid, MCA, University of El Oued, Supervisor

M. Ahcène Bounceur, MCA, University of Western Brittany, Co- Supervisor

Mr. Saber Benharzallah, Professor, University of Batna 2, Examiner

Mr. Messaoud Abbas, MCA, University of El Oued, Examiner

Mr. Abdelkamel Ben Ali, MCA, University of El Oued, Examiner

Mr. Kamal Amroun, MCA, University of Bejaia, Invited

N° de série :

République Algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITÉ ECHAHID HAMMA LAKHDAR D'EL OUED
FACULTÉ DES SCIENCES EXACTES
DEPARTEMENT D'INFORMATIQUE

Thèse de Doctorat

Domaine : Mathématiques et Informatique
Filière : Informatique
Spécialité : Systèmes d'Information avancés

Titre

La gestion sécurisée des entités cloud autonomes dans un système distribué

Par : Kara Mostefa

Soutenu le : 22/ 12/ 2022

Devant le jury composé de :

Mr. Kholladi Mohamed-khireddine, Professeur, Université d'El Oued, Président
Mr. Laouid Abdelkader, MCA, Université d'El Oued, Rapporteur
M. Bounceur Ahcène, MCA, Université de Bretagne Occidentale, Co-Rapporteur
Mr. Benharzallah Saber, Professeur, Université de Batna 2, Examinateur
Mr. Abbas Messaoud, MCA, Université d'El Oued, Examinateur
Mr. Ben Ali Abdelkamel, MCA, Université d'El Oued, Examinateur
Mr. Amroun Kamal, MCA, Université de Béjaia, Invité

acknowledgments

First of all, I thank ALLAH for having given me the strength and the patience necessary to complete this work.

Afterward, I would like to express my deep gratitude to my thesis director, Dr. Abdelkader Laouid who directed me at the beginning of the work to take the correct and short path to reach the goal, for having benefited me from his knowledge, for his advice, his patience, his availability and his support throughout these three years.

I also address my warmest thanks to my thesis co-supervisor Dr. Achène Bounceur, an Associate Professor at the University of Western Brittany; and my colleague Prof. Mohammad Hammoudeh, a Professor at King Fahd University of Petroleum and Minerals who guided and framed me with permanent enthusiasm.

I would also like to thank all the members of the jury: Prof. Mohamed-khireddine Kholladi, Professor at the University of El Oued, for having done me the honor of chairing my thesis jury, as well as Prof. Saber Benharzallah, Professor at the University of Batna 2, Dr. Messaoud Abbas, (MCA) from the University of El Oued, Dr. Abdelkamel Ben Ali (MCA) from the University of El Oued, and Dr. Kamal Amroun (MCA) from the University of Béjaia for having done me the honor of agreeing to judge this thesis and the interest they have shown in my work.

I would like to sincerely thank all the people who, through their encouragement, contributed to the completion of my doctoral thesis.

Finally, my thoughts go out to my loved ones without whom I will, of course, never have arrived here. I think of my parents, my family, my brothers, and my sisters.

Mostefa Kara

Contents

List of Figures

List of Tables

Abstract

The secure management of autonomous cloud entities in a distributed system

With the advances in technology and the abundance of information, cloud computing services have become indispensable because they enable the backup and availability of data, which implies access anytime and from anywhere. The collective use of cloud data poses an important issue in the security field, mainly in encryption to exchange or store data. Nowadays, cloud computing offers a suitable platform for users to exploit data. Nevertheless, the cloud can be an untrusted party exposing the decrypted data to misuse. Cryptography helps solve many computing problems, including securing the communication of cloud entities. For this reason, researchers are constantly working to find solutions to various challenges to preserve data and systems. Homomorphic encryption is an aspect of autonomy that enables secure use of the cloud. This work proposes powerful and efficient solutions represented by homomorphic encryption techniques.

In addition, homomorphic encryption cannot answer other questions for cloud entities. Cloud computing suffers from other issues such as lack of transparency, trust, and centralization of control. Blockchain has powerful features like decentralization that allow it to add enhancements to cloud computing, including autonomy, security, reliability, and transparency. Therefore, we propose a few consensus algorithms which are the hearts of the secure blockchain.

Keywords: Security, Cryptography, Homomorphic, Blockchain, Cloud.

Résumé

La gestion sécurisée d'entités cloud autonomes dans un système distribué

Avec les progrès de la technologie et l'abondance d'informations, les services de cloud computing sont devenus indispensables parce qu'ils permettent la sauvegarde et la disponibilité des données, ce qui implique un accès à tout moment et de n'importe où. L'utilisation collective des données cloud pose une issue importante dans le domaine de la sécurité, principalement dans le domaine du chiffrement pour échanger ou stocker des données. De nos jours, le cloud computing offre une plate-forme appropriée aux utilisateurs pour exploiter les données. Néanmoins, le cloud peut être une partie non fiable qui peut exposer les données décryptées à des abus. La cryptographie aide à résoudre de nombreux problèmes informatiques, y compris la securisation de communication des entités cloud. Pour cette raison, les chercheurs travaillent constamment à trouver des solutions à divers défis afin de préserver les données et les systèmes. Le chiffrement homomorphe est un aspect de l'autonomie qui permet une utilisation sûre du cloud. Ce travail propose des solutions puissantes et efficaces représentées par des techniques de chiffrement homomorphe.

En plus, d'autres questions pour les entités cloud ne peuvent pas être répondus par le chiffrement homomorphe. Le cloud computing souffre d'autres problèmes tels que le manque de transparence et de confiance ainsi que la centralisation du contrôle. La blockchain possède des fonctionnalités puissantes comme la décentralisation qui lui permettent d'ajouter des amélorations au cloud computing, notamment l'autonomie, la sécurité, la fiabilité et la transparence. Par conséquent, nous proposons quelques algorithmes de consensus qui sont les cœurs de la blockchain sécurisée.

Mots clés: Sécurité, Cryptographie, Homomorphe, Blockchain, Cloud.

ملخص

الإدارة الآمنة للكيانات السحابية المستقلة في نظام موزع

مع تقدم التكنولوجيا وكثرة المعلومات، أصبحت خدمات الحوسبة السحابية لا غنى عنها لأنها تتيح النسخ الاحتياطي للبيانات وتوافرها، مما يعني الوصول في أي وقت ومن أي مكان. طرح الاستخدام الجماعي للبيانات السحابية قضية مهمة في مجال الأمن، لا سيما في مجال التشفير لتبادل البيانات أو تخزينها. في الوقت الحاضر، توفر الحوسبة السحابية منصة مناسبة للمستخدمين لاستغلال البيانات. ومع ذلك، يمكن أن تكون السحابة طرفًا غير موثوق به يمكن أن يعرض البيانات التي تم فك تشفيرها لسوء الاستخدام. يساعد التشفير في حل العديد من مشكلات الحوسبة، بما في ذلك تأمين اتصالات الكيانات السحابية. لهذا السبب، يعمل الباحثون باستمرار لإيجاد حلول لمختلف التحديات من أجل الحفاظ على البيانات والأنظمة. التشفير متماثل الشكل هو جانب من جوانب الاستقلالية التي تتيح الاستخدام الآمن للسحابة. يقترح هذا العمل حلولاً قوية وفعالة تتمثل في تقنيات التشفير المتماثل.

بالإضافة إلى ذلك، لا يمكن الإجابة على الأسئلة الأخرى الخاصة بالكيانات السحابية عن طريق التشفير المتماثل. تعاني الحوسبة السحابية من مشكلات أخرى مثل الافتقار إلى الشفافية والثقة بالإضافة إلى مركزية التحكم. تتميز البلوكشاين بميزات قوية مثل اللامركزية التي تسمح لها بإضافة تحسينات إلى الحوسبة السحابية، بما في ذلك الاستقلالية والأمان والموثوقية والشفافية. لذلك، نقترح بعض خوارزميات الإجماع التي هي قلب البلوكشاين الآمن.

الكلمات المفتاحية: الأمان، التشفير، التماثل، البلوكشاين، السحابة.

Introduction

The Internet revolution and the increasingly massive creation of data in digital form facilitate communications and exchanges and therefore weaken the information stored. Indeed, public networks including distributed systems and cloud computing create security breaches also in terms of privacy and it is more manageable for an attacker to access data. Similarly, the replacement of human beings by machines and programs makes relations much more anonymous even though access to this information requires strong authentication methods.

In addition, the dematerialization and creation of other means of legal proof such as digital signatures instead of handwritten ones, all contributed to the increase in these security breaches. Thus, the digital revolution in the world of communication and information has opened many areas of security investigation as our daily lives have been invaded by smart cards, banking transactions, internet, mobile phone, etc.

There are many techniques to combat attacks targeting information in general and personal life in particular. These techniques are distributed on different levels such as networks, applications, storage devices, etc. However, one of the most important means of protection remains to keep this data with its original owner. The great development of technology such as IoT, cloud, speed of the internet, and the opening of the world to some of it led to the production of a huge amount of data, which led to the need to store data in a place other than the place of its production, we mean usually the Cloud.

When this data is personal or sensitive, the cloud cannot be trusted and the client can not store data in a readable way (in clair), so the client will have to encrypt it. On the other hand, the client not only uses storage service but may ask the cloud to perform operations on this encrypted data such as addition and multiplication. It is obvious to know that not any encryption method will allow these operations to be performed, meaning that the cloud entities will ask the client to decrypt the data before performing the operation, and this is exactly what happens with the classic encryption methods. That is, the cloud entities are not autonomous, and therefore privacy cannot be preserved.

In this work, we want to add autonomy to the cloud entities and make them process the encrypted data without the need to decrypt it by its original owner, which is called homomorphic encryption.

On the other hand, distributed systems may suffer, whether at the level of IoT or at the level of cloud entities, from some security problems that homomorphic cannot solve, such as domination, when these systems depend on the current leading technology in collecting and storing data, which is the blockchain. Therefore, we have developed consensus algorithms that are more robust against these attacks, knowing that consensus algorithms are the heart of the blockchain that relies on for collecting data.

Distributed systems

Perhaps the most important thing in our time is the information that has become in many forms, with its increasing size and because of the development of many technologies (LANs, WANs, Nano computers, etc.) it has become easy and necessary at the same time to create a computing system consisting of many elements connected to the network. These elements are usually geographically dispersed, which makes us say that they constitute a distributed system.

Many definitions of distributed systems have been presented in the literature. A distributed system is a set of autonomous units that appears to users as a single system [1]. This definition gives two characteristic features of distributed systems. The first is independence, that a distributed system is a set of units that can each operate independently of the other. The computing unit will generally be called the node (computers, physical servers, virtual machines, containers, software processes, etc.). A second element is that users (whether human, machine, application, etc) think they are dealing with a single system. So, the autonomous nodes have to collaborate (synchronization, concurrency, etc.) without considering any assumptions about the type of nodes and also no assumptions about how the nodes are interconnected.

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).

Cloud computing

Historically, distributed computing has been expensive, complex to process, and challenging to manage. After the appearance of cloud computing which offers extended functionalities, distributed computing has become more flexible and more affordable for users (customers). There was no specific date when we could say that cloud computing appeared [2]. Towards the end of the 90s, the concept of SaaS (software as a service) took on great importance with the appearance of grid computing [3] and the evolution of virtualization technology which was in the works of 'IBM [4].

The concept of cloud computing has its origins in the field of computing in 2006 [5] with the creation of Amazon EC2 (Elastic Compute Cloud). Then, in 2009, it was the time of the explosion of the cloud with the arrival of large companies such as IBM Smart Business Service (from IBM), Google App Engine (from Google), Microsoft Azure (from Microsoft), Sun Cloud (from Sun) and Ubuntu Enterprise Cloud (from Canonical Ltd). Currently, the notion of cloud computing has become well known in computer science. It is very common to hear about cloud drives, cloud databases, cloud servers, cloud security, etc. So, cloud computing is one of the most critical environments in the IT industry. There are several definitions for this concept, according to Microsoft Azure [6], "The cloud is the provision of computing services, including servers, storage, analytic, security, networking, software, and intelligence, etc. to deliver flexible resources, faster innovation, and economies of scale".

To meet various business needs, there are basic requirements of a cloud environment such as adaptability, extensibility, scalability, and manageability. It is found that the biggest change that cloud computing has to offer to compute systems and distributed systems is to move the data center offsite (i.e., to a third party) and to purchase services instead of dealing directly with applications onsite. Through the use of the cloud, companies can reduce operational costs and treatment time. Therefore, companies will focus on their main business instead of worrying about different IT issues. Thanks to the cloud, applications and documents are allowed to access from anywhere in the world using the Internet.

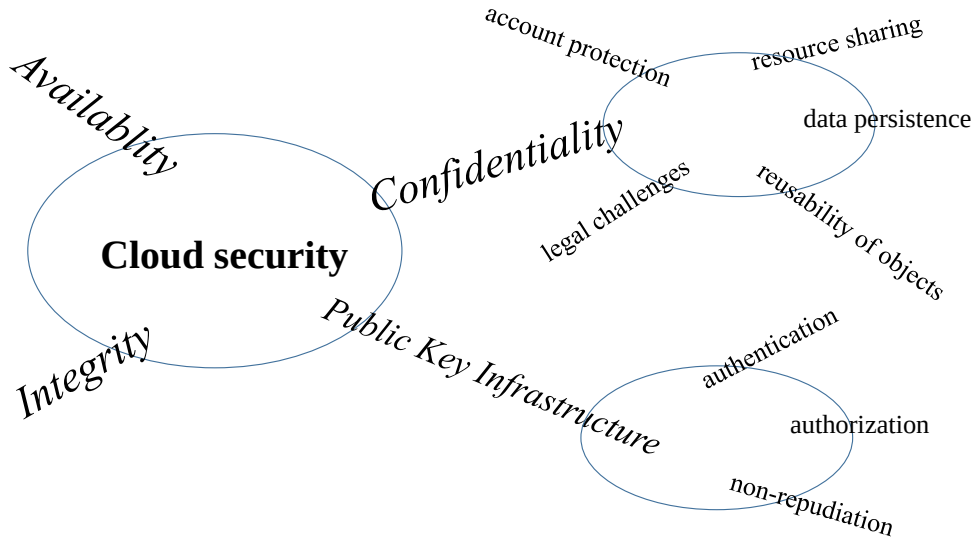

Figure 1: Aspects of security in a cloud environment

Security

When talking about information, we must not neglect an essential aspect, which is the security of data (including Privacy and Trust). Security can be defined by its objectives (within a distributed system), which consist of ensuring the confidentiality, availability, and integrity of data communicated or held in the participating systems. Second, Controlling access to services or their components. Third, there is a separation of data and processes at the virtual cloud level, to ensure no data leakage between applications. The security, in general, is related to three aspects, confidentiality, integrity, and availability. These aspects apply to the three major categories of assets that must be protected, data, software, and hardware resources (Figure 1).

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.

In a distributed environment, certain security problems are solved via a trusted third party which is an entity facilitating secure dealings of several users who trust it (end-to-end security services). This entity relies on PKI (Public Key Infrastructure). PKI offers technically sound and legally acceptable means to implement the following points:

- Authentication: identification of parties involved in transactions.
- Authorization: roles and authorization rights.
- Data confidentiality: ensuring authorized access.
- Data integrity: avoiding unauthorized modifications.
- Non-repudiation: it guarantees that no user can deny its presence in a deal.

Autonomous environment

Among the thing that can ensure security is autonomy. An autonomous system is capable of achieving a set of objectives in a changing environment, by observing, acquiring then reacting (according to rules) for an extended period of time without human control or intervention. Common examples include driverless cars, mobile agents, and autonomous mobile robots.

An autonomy system is able to do the following actions:

- Observe the environment and keep track of the current status.
- Perceive and understand different data sources.
- Decide the following action to do and develop a plan.
- Acting only when it can do so safely, avoiding unsafe situations.
- Doing tasks without the need for human intervention.

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.

Principal contributions

After talking about all this, the factor that presents itself here is security. The question that comes to mind is: Does the cloud achieve security for its users? The answer to this question is what we will achieve in this research, first by securing the management of autonomous cloud entities. This is realized by homomorphic encryption to perform arithmetic operations on the data stored in the cloud. Secondly, by giving algorithms that guarantee the collection and preservation of data in a secure manner. Therefore, we have the following contributions:

1. We propose a fully homomorphic encryption technique by using twin key and magic number fragmentation. To ensure the homomorphic addition and multiplicative properties.
2. We propose an asymmetric homomorphic encryption technique that is based on distributing a number's digits to key in such a way that Additive Homomorphic Encryption is performed.
3. A consensus algorithm that ensures data reliability and integrity in untrusted environments. We improve the Proof of Work (PoW) consensus algorithm by introducing many proof rounds reinforcing security against attacks.
4. Other improvement in consensus algorithms consist of proposing Delegated Proof of Work (DPoW) reinforcing security against the 51% attack.

Structure of the manuscript

The following Chapter is dedicated to preliminaries when we will show math tools, cryptography, cryptosystems, key exchange protocols, cloud computing, and blockchain. In Chapter 2, we will talk about the homomorphic encryption for cloud computing environment providing the proposed techniques, performance, security analysis, and Implementation. In Chapter 3, we will discuss blockchain for security concerns including proposed protocols, efficiencies, cryptanalytic attacks, and experimental results. Finally, we conclude with a conclusion.

List of publications

Journal papers:

1. A Fully Homomorphic Encryption based on Magic Number Fragmentation and ElGamal Encryption: Smart Healthcare Use Case, Experts systems journal, 2021 [7].

   Authors: Mostefa Kara, Abdelkader Laouid, Mohammed Amine Yagoub, Reinhardt Euler, Saci Medileh, Mohammad Hammoudeh, Amna Eleyan and Ahcène Bounceur.

2. A Compute and Wait in PoW (CW-PoW) Consensus Algorithm for Preserving Energy Consumption, Applied Sciences journal, 2021 [8].

   Authors: Mostefa Kara, Abdelkader Laouid, Muath AlShaikh, Mohammad Hammoudeh, Ahcene Bounceur, Reinhardt Euler, Abdelfattah Amamra and Brahim Laouid.

3. Secure Key Exchange Against Man-in-the-Middle Attack: Modified Diffie-Hellman Protocol, Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, 2021 [9].

   Authors: Mostefa Kara, Abdelkader Laouid, Muath AlShaikh, Ahcène Bounceur and Mohammad Hammoudeh.

4. Perfect Confidentiality through Unconditionally Secure Homomorphic Encryption Using One-Time Pad with a Single Pre-Shared Key, Journal of Information Science and Engineering, 2022 [10].

   Authors: Mostefa Kara, Abdelkader Laouid, Ahcène Bounceur, Mohammad Hammoudeh and Muath AlShaikh.

5. Proof of Chance: A Lightweight Consensus Algorithm for the Internet of Thing, IEEE Transactions on Industrial Informatics, 2022 [11].

   Authors: Mostefa Kara, Abdelkader Laouid, Mohammad Hammoudeh, Muath AlShaikh, and Ahcène Bounceur.

Conference papers:

1. A Homomorphic Digit Fragmentation Encryption Scheme Based on the Polynomial Reconstruction Problem, ICFNDS '20: The 4th International Conference on Future Networks and Distributed Systems - St. Petersburg Russian Federation, November, 2020 [12].

   Authors: Mostefa Kara, Abdelkader Laouid, Reinhardt Euler, Mohammed Amine Yagoub, Ahcène Bounceur, Mohammad Hammoudeh and Saci Medileh.

2. A Novel Delegated Proof of Work Consensus Protocol, International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), November, 2021 [13].

   Authors: Mostefa Kara, Abdelkader Laouid, Ahcene Bounceur, Farid Lalem, Muath AlShaikh, Romaissa Kebache and Zaoui Sayah.

3. Semi-Decentralized Model for Drone Collaboration on Secure Measurement of Positions, ICFNDS '21: The 5th International Conference on Future Networks and Distributed Systems Dubai, UAE, December, 2021 [14].

   Authors: Mostefa Kara, Abdelkader Laouid, Ahcène Bounceur, Mohammad Hammoudeh, Muath AlShaikh and Romaissa Kebache.

4. Secure Clock Synchronization Protocol in Wireless Sensor Networks, NCASAM21: 1st National Conference on Applied Science and Advanced Materials, Skikda, Algeria, December, 2021 (accepted).

   Authors: Mostefa Kara, Abdelkader Laouid, Ahcene Bounceur and Mohammad Hammoudeh.

5. Secure Consensus Clock Synchronization in Wireless Sensor Networks, International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), November, 2021 [15].

   Authors: Aissaoua habib, Abdelkader Laouid and Mostefa Kara.

6. A Multi-Key Based Lightweight Additive Homomorphic Encryption Scheme, International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), November, 2021 [16].

   Authors: Khaled Chait, Abdelkader Laouid, Lamri Laouamer and Mostefa Kara.

7. Asymmetric Image Encryption Based on Twin Message Fusion, International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), November, 2021 [17].

   Authors: Mohammed Elhabib Kahla, Mounir Beggas, Abdelkader Laouid, Mostefa Kara and Muath AlShaikh.

Communications:

1. Secure management of autonomous cloud entities in a distributed system, IPPM'20: International Pluridisciplinary PhD Meeting (IPPM'20), University of El-Oued, February, 2020.

   Author: Mostefa Kara

2. A Lightweight Clock Synchronization Technique for Wireless Sensor Networks: A Randomization-Based Secure Approach, CNTA'21: National Conference on Telecommunications and its Applications, Ain-Témouchent, Algeria, December, 2021.

   Author: Mostefa Kara

Chapter 1

Preliminaries

1.1 Introduction

We live in the age of the cloud, an ambiguous entity with the power to store and process data, accessible from anywhere in the world via an internet connection. The cloud materializes in a multitude of servers that are located somewhere on the planet. The cloud is a very practical environment; all you need is a connection device to access your calendar, photos, documents, contacts, emails, etc. Several IT services are offered by different companies (cloud providers) and most of the time their products are free. Of course, nothing is free even with the cloud. It is true that our data seems protected by a password that only we have, but the cloud service provider is the absolute final recipient of all the data.

Accessing this data gives the cloud an enormous amount of very valuable detail about ourselves. This third party knows our tastes and needs, and it can resell all this information. So we finally became the product to exploit. In this scenario, it becomes essential to find practical solutions to ensure the protection of private data, without at the same time losing the advantages brought by the cloud. This thesis presented a set of security solutions at different levels; in this chapter, we will see some concepts that pave the way for that, including encryption, cloud computing, and blockchain.

1.2 Maths tools

This section makes a reminder of some mathematical tools that we need to talk about security in general and more specifically encryption.

1.2.1 Definitions and notations

$\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z} = \{0, 1, \ldots, p-1\}$,
Key : it is the parameter involved and authorizing encryption and/or decryption operations.
pk : the public key is a large numerical value that is used for encryption and checking the legitimacy of a digital signature.
sk : in a symmetric cryptosystem, the secret key is used for encryption and decryption of data, in an asymmetric cryptosystem, the private key is used for decryption data that were encrypted by the corresponding public key or to create signatures.
m : plain text,
c : cipher text,
P : the ring of plain text $\mathbb{Z}_p$,
C : the ring of cipher text $\mathbb{Z}_n$,
K : the ring of key,
$Enc(m)$: the encryption function, to convert a message into an unreadable format, a plain text $m \in P$ is encrypted using the public key $Enc_{pk} : P \longrightarrow C$ with $pk \in K$,
$Dec(c)$: the decryption function, to return the original message; a cipher text is decrypted using the secret key $Dec_{sk} : C \longrightarrow P$ with $sk \in K$.
$KeyGen$: that returns a secret (or private) and public key, respectively sk and pk,
mod : modular arithmetic for integers is an operation that considers the remainder. In which, numbers will wrap around upon reaching a given limit (this limit is known as the modulus) to leave a remainder. Modular arithmetic is often used in prime numbers. $r = a \mod b$, a is the dividend, b is the divisor, and r is the remainder.
log_b : logarithm of base b,
$sha256$: the hash function is a one-way mathematical function that uses inputs of variable lengths and returns outputs of a fixed length. In this function, it must be hard to guess the input value from its output. Moreover, it must be hard to select an input that provides a pre-defined output.
$\times$: multiplication operator,
$+$: addition operator,
$(\frac{a}{b})$: is the quotient of $a \div b$.
$\sum_{i=1}^{t} m_i$: denotes the sum, $m_1 + m_2 + \cdots + m_t$.
$\prod_{i=1}^{t} m_i$: denotes the product, $m_1 \times m_2 \times \cdots \times m_t$
$Cryptology$: it is a mathematical science with two branches: cryptography and cryptanalysis.
$Cryptography$: the study of methods giving the possibility of sending data confidentially on a given medium.
$Prime\ number$: a prime number p is a whole number realizes $p \geq 2$, and the only factors of p are 1 and itself.

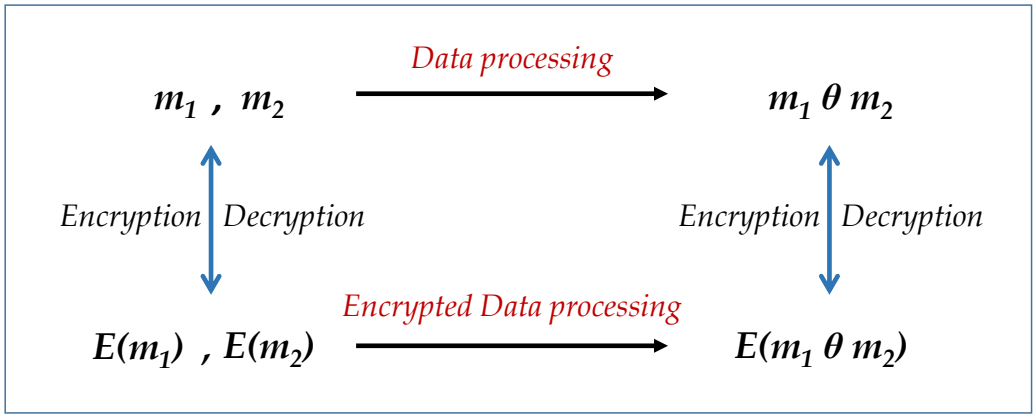

Figure 1.1: Homomorphic encryption

Cryptanalysis : opposed to cryptography, its purpose is to find the plain text from ciphertexts by determining the flaws of the algorithms used.
Encryption : it consists of transforming data (text, number, etc.) in order to make it incomprehensible to persons other than that who created the data and the person to receive it.
Cryptosystem : it is defined as the set of possible keys (keyspace), possible plain, and ciphertexts associated with a given algorithm.
Kerckhoff's principle : the security of the cipher should not depend on what cannot be easily changed. "The security of an encryption system should only be based on the secrecy of the key".
Homomorph : homomorphism is a special correspondence between the elements of two algebraic systems including two groups, rings, or fields. Two homomorphic systems have the same basic structure, and, while their elements and operations may appear entirely different, results on one system often apply as well to the other system. Therefore, if a new system can be shown to be homomorphic to a known system, certain known features of the first can be applied to the second, thereby simplifying the analysis of the new system.

So, Homomorph is a function has a special algebraic property which is the transformation from a data-set M to another C with preserving a special correspondence between each two elements from M and C respectively. Homomorphic encryption enables any third party to perform computational operation on encrypted data without decrypting them beforehand as shown in Figure 1.1 where θ denotes an algebraic operation. So, $Enc(m_1 + m_2)$ is easy to compute from $Enc(m_1)$ and $Enc(m_2)$.

Anonymization : consists of modifying the content or structure of data in order to make it very difficult or impossible to "re-identify" the persons (natural or legal) or entities concerned.

$TTPTrustedThird - Party$: is an entity that facilitates secure interactions between two parties who trust this third party. A trusted third party is a natural or legal person authorized to carry out legal security operations of authentication, transmission and storage. A responsible third party that all participants agree on in advance to provide a service or function such as certification.

Substitution : is a classic encryption, it consists of replacing each letter with a different letter, for example Caesar cipher (50 BC), Vigenera system (1586), and Verman's system (1917).

1.2.2 Problems related to factorization

Given an integer N, calculate non-trivial factors of N (different from 1 and N). This problem is easy if N has a special form, for example even. Hard instances of this problem seem to be numbers of the form: $N = p \times q$ where p and q are two large prime numbers of equivalent size. We can therefore conjecture that the function $f(p, q) = p \times q$ is one-way: i.e., it is easy to compute $N = p \times q$, given p and q, in time $O(k^2)$, but there does not exist a probabilistic algorithm in polynomial time which, on a random input N of the form $p \times q$, makes it possible to calculate p and q in reasonable time on average for pairs (p, q) chosen at random among the large prime numbers. Similarly, it is easy to generate p and q and test their primality in $O(k^3 \times log(k))$ time. The problem underlying factorization is the RSA problem. Given (N, e, y) where $e, y \in Z$ and $N = p \times q$, find x such that $y = x^e \mod N$.

1.2.3 Problems related to the computation of the discrete logarithm

Let g be a generator of order q of a subgroup $G \subset Z$. By Lagrange's theorem, q is a divisor of $p1$ because p is prime [18]. Given p, g, and $y \in G$, find x such that $y = g^x \mod p$. The so-called discrete logarithm hypothesis is therefore that it is difficult to compute x in a reasonable time if y is a random instance in Z. The numbers p and g are not necessarily chosen at random. The prime number p is selected to have certain properties that make it difficult to calculate the discrete logarithm. For example, p must be chosen such that $p - 1$ has large prime divisors. The problem underlying discrete logarithm is the Calculatory Diffie-Hellman problem. Given g, $g^x \in G$ and $g^y \in G$, calculate $g^{x \times y}$

1.2.4 Problems used for the homomorph

There are several problems used for the homomorph [19], we will be interested in the following:
Approximate Greatest Common Divisor problem: The security of modern cryptographic primitives is generally based on mathematical problems considered challenging to solve. These problems are many, but the approximate common divisor (ACD) problem is attractive because it is easy to manipulate and describe, it relies only on simple operations on integers. Even the probability distributions used in its definition are elementary since it only uses the uniform distribution (instead of, for example, the discrete Gaussian distribution). Although simple, the ACD problem (also known as the AGCD problem) has been widely used to construct advanced primitives such as fully homomorphic encryption (FHE).

Let a secret integer p then samples several random integers q_i's and defines multiples of p as: $m_i = p \times q_i$. Now, given these m_i's, of course, it is easy to recover p: we can simple compute the greatest common divisor, that is, $p = gcd(m_1, m_2, \dots m_t)$. But what if we are given "approximate multiples" of p instead of exact multiples, that is, if one adds small integers r_i's to the m_i's? Given values of the form $x_i = p \times q_i + r_i$, how can we recover p?

1.3 Cryptography

Created thousands of years ago, cryptography is now ubiquitous to protect our digital data, especially in distributed systems and cloud environment [20, 21].

1.3.1 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 [22]. 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.

1.3.2 Symmetric scheme

Symmetric encryption, also called secret key encryption when Alice and Bob use the same key to both encrypt and decrypt operations [23, 24, 25, 26]. For example, Rijndael (AES) [27, 28] presented a multi-turn block cipher with larger and variable block and key sizes. Hill's work [29, 30] uses modulo 26 to encrypt the 26 letters.

A symmetric encryption scheme [31] 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:

$$c = E(m; k) = E_k(m) \tag{1.1}$$

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

$$m = D(c; k) = D_k(c) \tag{1.2}$$

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 1.2 shows the scenarios and the consequences of using symmetric encryption.

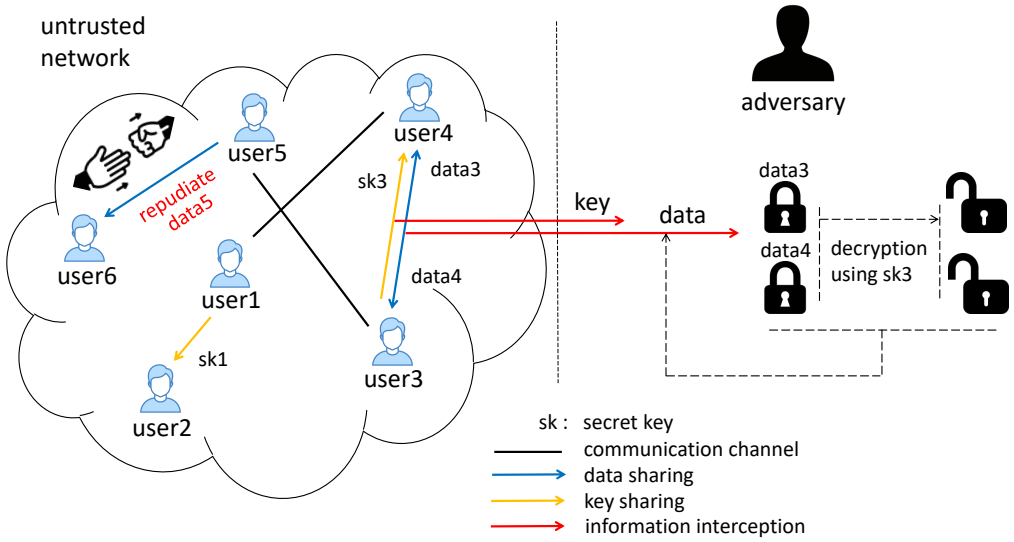

Figure 1.2: Overview of symmetric encryption

1.3.3 Asymmetric scheme

The public-key cryptosystem introduced by Diffie-Hellman [21] allows encryption from the recipient's public key. Only the latter, equipped with a private key associated with his public key, can decrypt the message [32, 7, 12]. Public-key cryptography is sometimes called "asymmetric cryptography", 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. [33]. Asymmetric technique is based on the one-way functions i.e., it is simple to apply this function to plaintext, but extremely difficult to find this plaintext 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, 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. To ensure data protection and the unimpeded operation of asymmetric encryption, it is necessary to remain sk secret and is 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 turns them up to nearly a thousand times faster.

1.3.4 Use of cryptography

Cryptography has many uses, including encryption, entity authentication, data integrity, data origin ensuring, non-repudiation, etc. The symmetric algorithms are specially used for cipher block encryption, data integrity, confidentiality, and authentication. Asymmetric algorithms are generally used for ensuring symmetric key exchange and offering digital signatures. Certification authorities (such as a trusted third-party) are based on Asymmetric cryptography for the establishment of Public Key Infrastructures (PKI) in order to prevent impersonation attacks [34].

For more details [35]:

- **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. Figure 1.3 shows the threats of interception of information in transit or in storage.

- **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 ensured storage integrity with keys, locks, and other safeguards to avoid unauthorized and suspicious

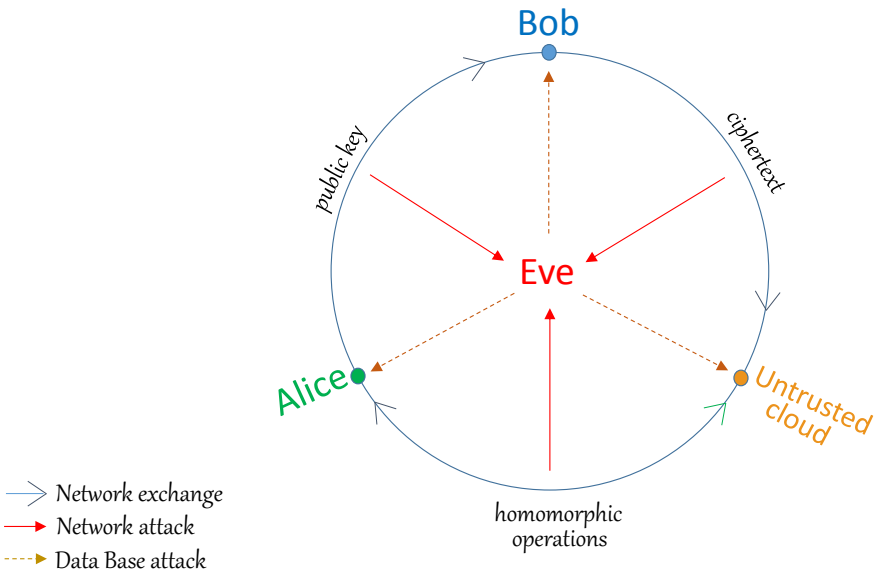

Figure 1.3: Interception attack in transmission and in storage

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 an 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, 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.

1.3.5 Quantum cryptography

Also known as quantum encryption or quantum security. A provably secure key agreement system has been suggested whose security is based on the Heisenberg uncertainty principle of quantum physics. The security of so-called quantum cryptography does not depend upon any complexity-theoretic hypotheses [36]. When elementary quantum systems, such as polarized photons, are used to transmit digital information, the uncertainty principle of quantum physics provides rise to novel cryptographic phenomena, unachievable with standard communication media. For instance, there is a communication channel on which it is impossible to

eavesdrop without a high probability of confusing the transmission in such a manner as to be detected. This encryption method takes advantage of basic physics laws, such as the observer effect, which says that it is impossible to identify the particle's location without changing that particle.

Such a channel offers one of the main benefits of public-key cryptosystems; it allows the secure distribution of critical data between two users who initially do not share any secret data, provided the users have access. For an ordinary channel exposed to passive eavesdropping but not active tampering, both users can still exchange a key securely if they initially share secret data [37]. Post-quantum cryptography refers to algorithms that are believed to have protective capabilities against an attack by a quantum computer.

1.3.6 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.

1.4 Cryptosystems

Although cryptography is effective in our systems, it contains certain limits, such as the modification of data by third parties who store it. These third parties must possess the client data in the clear before modifying it. This poses a problem of data confidentiality and client privacy. Homomorphic encryption is a solution to this problem. A homomorphic schema ensures data confidentiality, for such as a

cloud computing service. In other words, one of the main advantages of homomorphic encryption is being able to delegate our calculations to cloud servers without losing data confidentiality. There are four types of homomorphic cryptosystems.

1.4.1 Partially homomorphic cryptosystems

Partially Homomorphic Encryption (PHE) cryptosystem cover only one sort of operation, multiplication or addition, with an unlimited number of times [38]. As multiplicative schemes, we can find RSA [32] and El-Gamal [39] cryptosystems. As an additive cryptosystems, we can find Naccache and Stern 1998 [40], Paillier 1999 [41], Galbraith 2002 [42] and Kawachi et al. 2007 [43].

Consider the following example of an additive system,

$F(x) = 2 \times x$

the client encrypts the values 5 and 6, he obtains $c_5 = 10$ and $c_6 = 12$, the results are stored in the cloud,

if the client wants to calculate the sum of these two numbers (5 and 6), the cloud does this calculation and returns the result to the client, $r = c_5 + c_6 = 22$,

the client just decrypts r to get the result of adding 5 and 6,

$F^{-1}(x) = x/2$

so $F^{-1}(22) = 11 = 5 + 6$

1.4.2 Leveled homomorphic cryptosystems

Leveled Homomorphic Encryption (LHE) schemes support both addition and multiplication with a limited number of times. Indeed, LHE supports finite operations over ciphertexts mainly from the perspective of circuit depth. The circuit depth will be predetermined in the setup algorithm [44, 45, 46].

In other words, if a system, for example, does only one addition operation on the ciphertexts, then the decryption of the result of the second addition operation will give an invalid result.

Consider the encryption function F of which: $F(x) = x_1$; so the cloud can only do $r = x_1 + y_1$ and the client gets $F^{-1}(r) = x + y$.

but with $r = (x_1 + y_1) + z_1$, $F^{-1}(r) \neq x + y + z$

1.4.3 Somewhat homomorphic cryptosystems

Many references consider LHE and Somewhat Homomorphic Encryption SHE to be the same thing, they call this type the same name. But in this manuscript, we want to do more detail since there are two parts to this type of encryption.

Like LHE, SHE covers both addition and multiplication operations but not together. 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, a SHE scheme allows us to do several additions or several multiplications without merging them [47, 48, 49].

1.4.4 Fully homomorphic cryptosystems

Fully Homomorphic Encryption (FHE) allows operations (both addition and multiplication) on ciphertexts with an unlimited number of times. FHE performs an absolute combination of operations with arbitrary functions, which gives a great advantage and strength to this type of encryption [50, 51, 52, 53, 54].

1.5 Key exchange protocols

A key exchange protocol (KEP) is a way that allows many users to agree on a cryptographic key. As shown in Figure 1.4, where f denotes the function that computes the output; noting that is not necessarily f_i equals f_{i+1}. After n iterations, the system provides the encryption key. The first key exchange protocol appeared thanks to asymmetric cryptography in the 1970s. KEP provides a secure use even with an insecure channel of communication, where a trusted third party is not used. Before the advent of public-key cryptography (asymmetric cryptography), there were no efficient mathematical methods for being able to exchange keys; The reliance entirely on operational security was that the key had to be passed over a truly secure channel; In other words, the user must rely on the physical transfer of diplomatic bags containing the keys.

The first key exchange technology was introduced at [55]. The idea was to develop a set of problems (puzzles) that require a certain effort to find the appropriate solution, and two secrets are revealed once they are solved. The user randomly chooses one of these puzzles and solves it in order to obtain a key. After that, the first secret is transmitted to the puzzle owner and notified of any puzzle that has been solved, where the second secret is the key. The attacker who listened to the conversation cannot know in advance which puzzle was chosen. To find

the key chosen by both users, the attacker has to solve all the puzzles presented. Therefore, the attacker solves the quadratic complexity problem. But creating a set of puzzles each time is not a practical method at all. Also, the Oracle stochastic model shows that the security of this technique is actually quadratic in the number of puzzles. This level of complexity is not considered sufficient in modern cryptography, so in [56] this technique shows that [55] technology cannot be improved.

The Diffie-Hellman technique only gives security against a passive adversary, when the adversary can overhear the conversation but cannot interrupt it. If the adversary can intercept data, he can perform a man-in-the-middle attack. This enables him to impersonate one of the interlocutors. So, he will create a key with each user. Finally, he will be able to decipher the data sent by one to the other. In 2002, [57] demonstrated the use of Weil couplings on elliptical curves to create a three-party key exchange. Using the Diffie-Hellman protocol requires the establishment of a safe channel between every two users. In [58], the authors proposed a generalization of [57] to ensure key exchange between an arbitrary number of users, employing a cryptographic multilinear application.

To ensure the security of the key exchange by exploiting the properties of physics, quantum key exchange protocols have been proposed. More specifically, based on the statistical properties of a flow of entangled particles linked to the non-cloning theorem, it is easy to find an attacker who is observing a set of the bits. This can be revealed by correcting the noise. Many of these protocols have been proposed and implemented in distances covering a few hundred kilometers [59]. This kind of exchange makes technological and operational challenges, these issues prevent its use. In particular, a specific transmission channel must be installed between the parties. For the BB84, B92 [60], and SARG04 [61] protocols, it must ensure the transportation of low-energy photons with conserving their polarization regardless of decoherence phenomena over the entire length of the channel.

1.6 Cloud computing

Cloud computing has transformed the information technology delivery system from a product to a service. It has offered the availability of various applications, platforms, and infrastructure resources as scalable, on-demand services over the Internet. However, the functionalities and performance of cloud computing services are hampered due to their vulnerability and failures due to the domains where they operate. It is possible to use the performance of cloud services to their maximum potential only if cloud service providers effectively manage reliability, availability, and privacy issues [62].

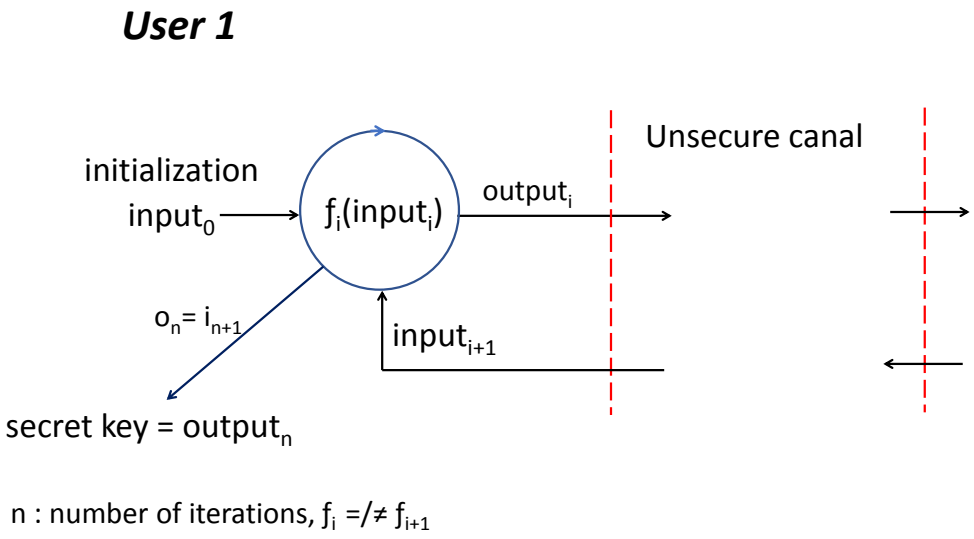

Figure 1.4: General diagram of key exchange protocol

Cloud computing refers to accessing, manipulating, and operating resources (such as software and hardware) at a remote location [63]. Buyya et al. [64] defined Cloud computing in terms of distributed computing "A Cloud is a type of parallel and distributed system containing a set of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers."

According to the U.S. National Institute of Standards and Technology (NIST) definition: "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (for example, servers, networks, storage, services, and applications) that can be quickly provisioned and released with least management effort or service provider interaction" [65].

Cloud computing offers various resources as services to on-demand clients. It allows companies and clients to use applications and software without installing them on physical machines and enables the use of required resources via the Internet. It offers features such as high computing performance, pay-as-you-go, connectivity, user interactivity, working reliability, ease of programming, efficiency, scalability, easy management of a lot of data, and the elasticity to turn IT from a product into a service [66, 67].

1.6.1 Characteristics of cloud computing

By its nature, which depends on several things, it contains many characteristics [68], including the following [2]:

- Free access on demand
- Service must be accessible via a network
- Shared resources
- Quick elasticity
- Metered service
- Pay-as-you-go
- Based Virtualization
- Based on SLA (Service Level Agreement)
- Simplicity, flexibility, reliability, and fault tolerance
- Effective security

1.6.2 Types of cloud services

The basic service models are:

- **Software-as-a-Service (SaaS):** In this type, software applications are offered by cloud service providers in the form of services to end clients. An application offered as a service to the user eliminates the need to install and run the cloud application on the user's local computer while avoiding tedious maintenance [69]. In this type, we can find web conferencing services, messaging apps, social media platforms, etc. In this type, we can see the following SaaS providers Amazon AWS, Google Compute Engine, Microsoft Azure, IBM SmartCloud Enterprise, CloudStack, OpenStack, OpenNebula, CloudForge, Citrix, Qstack, etc.
- **Platform-as-a-Service (Paas):** This type offers a platform to develop, run, test and manipulate applications in the cloud [70]. A client can rent an environment with a software stack and use it for custom application development. In this type, we can find the following PaaS providers: Acquia Cloud, Amazon AWS, App Agile, Apprenda, AppScale, Bluemix, Cloud 66, Cloudways, etc.

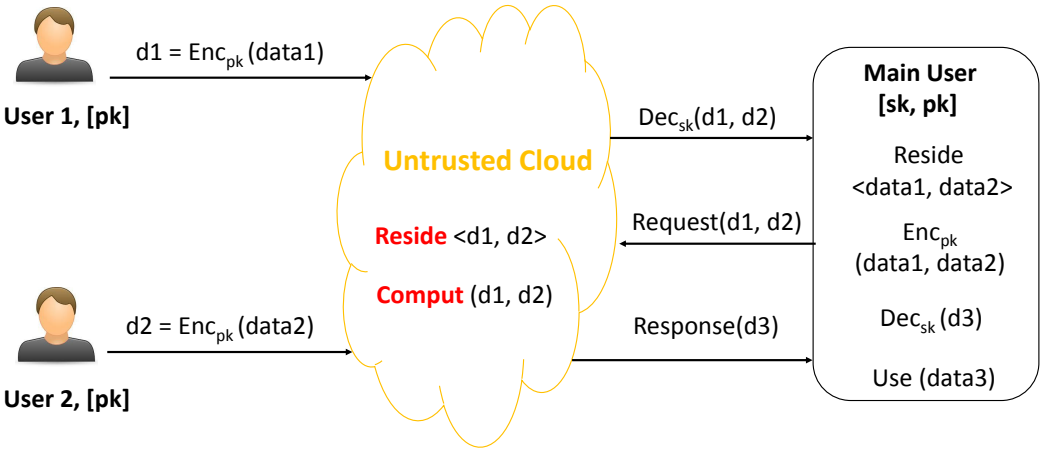

Figure 1.5: Assuring reside and compute by cloud

- **Infrastructure-as-a-Service (IaaS):** The IaaS type offers access to certain hardware resources, namely networks, physical machines, storage, servers, virtual machines on the cloud, etc. [71]. The IaaS provider offers services such as dynamic provisioning of virtual machines, installations, and on-demand storage. In this type, we can find the following SaaS providers Salesforce, Microsoft, Amazon Web Services, Slack, Zendesk, GitHub, Oracle, Cisco, etc.
- **Anything-as-a-Service (XaaS):** XaaS is another type of service that can be anything or everything as a service. The cloud environment can maintain considerable resources to meet client requirements such as Security-as-a-Service, Identity-as-a-Service, Database-as-a-Service, Communication-as-a-Service, Blockchain-as-a-Service, Strategy-as-a-Service, etc. [72].

Among cloud services, we are interested in security issues in computing and storage (Reside and Compute, Figure 1.5)

1.6.3 Cloud computing deployment models

The cloud deployment model is surely based on the motive and the environment in which the cloud services are supposed to be operated. The choice of the deployment model determines the cost incurred, the energy to be consumed by the

resources as well as the other capital expenditure [73]. The most well-known deployment models in cloud technology are public, private, community, and hybrid cloud.

- **Public Cloud:** The public cloud allows large users to access the services offered by the cloud provider. It offers flexibility, scalability, and geographical independence with a generally low cost due to the use of [74] multitenancy. Resources are dynamically provisioned at the request of clients from a remote tier that offers resources using a multi-tenant approach.
- **Private Cloud:** The private cloud is used within a particular organization, i.e., the resources and services offered by the cloud can be accessed or used just within the organization. This type provides high security and privacy at the application and data level [75].
- **Community Cloud:** This model is used simultaneously by various companies or organizations and helps interact with particular communities or companies that contain common implications (e.g., security necessities, mission, compliance considerations, etc.). This type may be operated, owned, or managed by one or more organizations within the community [76].
- **Hybrid Cloud:** The hybrid cloud is an alliance of two types, public cloud, and private cloud. In this model, critical events (for example, those requiring security) are realized using private cloud services, and non-critical events are realized using public cloud [77].

Public clouds are more suitable in scenarios where providers wish to offer collaboration services such as chat and video conferencing where there is insufficient resource availability and IT infrastructure locally. On the other hand, if security and privacy are very necessary, it should use a private deployment model. On the other hand, it should use a hybrid deployment model for an organization that already has a large IT infrastructure and is also expanding its capabilities.

1.6.4 Cloud computing security requirements

Cloud Computing security must also be guided to well-specified standards in order to become an effective and secure technological solution [2].

- Confidentiality: the information is known only to the communicating entities.
- Integrity: the information has not been modified between its creation and its processing.

- Availability: the information is always accessible, neither blocked nor lost.
- Identification and authentication: ensure the legitimacy of the access request made by an entity.
- Authorization: define the roles of each user.
- Trust: it describes whether an entity can be assumed to always behave according to protocol within the given model. A most common use of "trusted" in cryptography is to state that what's qualified as "trusted", by hypothesis, behaves, works, and performs per a particular precise set of rules (which depends on context), and will perform no less and no more. That's the case in a trusted third party and trusted server.
- Audit and compliance: a security audit is a systematic evaluation of the security of a company's information system by measuring how well it conforms to an established set of criteria.
- Non-repudiation: ensuring that the sender of data is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the data.

1.6.5 Security threats in cloud environment

The primary concern in cloud computing environments is to provide security around tenancy and computation which thus provides clients with more comfort and more confidence in the Cloud [78].

- **Basic security:** Web 2.0 is the main technology allowing the exploitation of software as a service (eg SaaS). This technology relieves clients of software maintenance and installation tasks. It has been widely used at all times and everywhere. As the community of users using Web 2.0 continues to grow rapidly, security and privacy have become more important than ever in such an environment [79, 80, 81]. In this context, we can find the following attacks: SQL injection attack, XSS attack, Man-in-the-middle attack, etc.
- **Application level security:** Application-level security refers to the use of software and hardware resources whose purpose is to secure applications so that attackers cannot have any access, control, or modification over the applications. Application security threats include XSS attacks, cookie poisoning, hidden field manipulation, SQL injection attacks, denial of service attacks, backdoor and debugging options, CAPTCHA breaking, etc. [82]

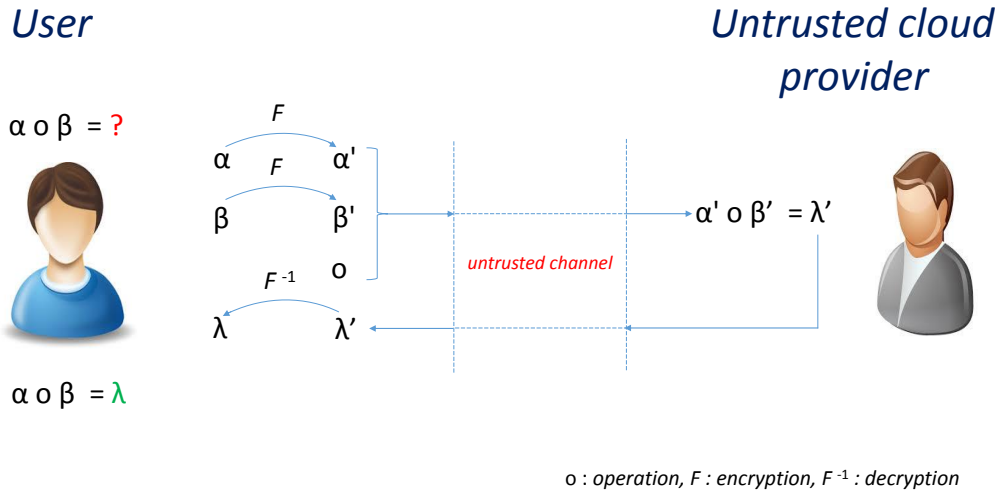

Figure 1.6: Untrusted channel

- **Physical security:** Physical security is more important than any other control to ensure cloud security. It threatens the premises or the building in which the Cloud Datacenters are hosted, such as human actions and natural hazards. Placing data centers in a disaster area is as dangerous as granting privileged access to all users or third parties. For this, the physical security of a room is a protection system that provides a multi-level or multi-layer defense, each layer of which is equipped with automated general control.

- **Data security:** Data protection in the cloud environment is the same as traditional data security, but due to the openness and multi-tenant characteristics of the cloud environment, the security here contains some peculiarities. The issue of data security and privacy is illustrated throughout the data lifecycle [83, 84]. The data lifecycle concerns the whole process from generation to destruction of data.

- **Network level security:** Networks or channels are divided into several types such as shared and non-shared networks, public and private, small and large networks, etc. Each of these types has to deal with a number of security threats. To ensure network security, it is necessary to consider factors such as network confidentiality and integrity, proper access control, and ensuring security against third-party threats, etc. (Figure 1.6). Network-level attacks include the following examples: DNS attacks, Sniffer attacks, IP address hacking or reuse, Denial of Service (DoS) attacks and Distributed Denial of Service (DDoS) attacks, etc.

1.7 Blockchain

Blockchain is a new technology for storing and transmitting information evolving without centralized control; it uses peer-to-peer networks and allows the constitution of replicated and distributed registers, these registers are secured thanks to cryptography by the linking of blocks to each other.

1.7.1 Definition

Blockchain is a distributed database in a network of nodes; it consists of approved transactions before they are recorded in a chain of computer code. Transaction details are recorded on a public ledger that anyone on the network can see [85].

1.7.2 Uses

Blockchain can be used for many different applications [86] by providing different services (Figure 1.7).

- **Cryptocurrencies:** Blockchain's most well-known use (and maybe most controversial) is in cryptocurrencies. Cryptocurrencies are digital currencies (or tokens) like Bitcoin, Ethereum, Litecoin, etc. the coins can be used to buy goods and services.
- **Financial Contracts:** The introduction of smart contracts in the blockchain enables the automation of many financial contracts on the blockchain [87, 88, 89]. Smart contracts are simply programs (small applications) stored on the blockchain that run when a set of predetermined conditions are met. They are typically exploited to automate the execution of an agreement between a set of participants. Finally, all participants can be immediately certain of the result, of course without the intervention of an intermediary or loss of time. The examples of smart contracts on a blockchain are numerous such as loans, insurance, etc.
- **Asset Tracking:** Another important use case for blockchain is as an asset tracking tool to verify proof of ownership as well as the provenance of a particular asset [90, 91].
- **Digital Identity:** Just as blockchain can be used to track ownership and provenance of goods, it can also be used to store the identity of people [92, 93, 94]. If the passport is stored on a blockchain, the visas, the entry, and departure from countries are recorded as blockchain transactions. This means that they are immutable, community-verified, and decentralized.

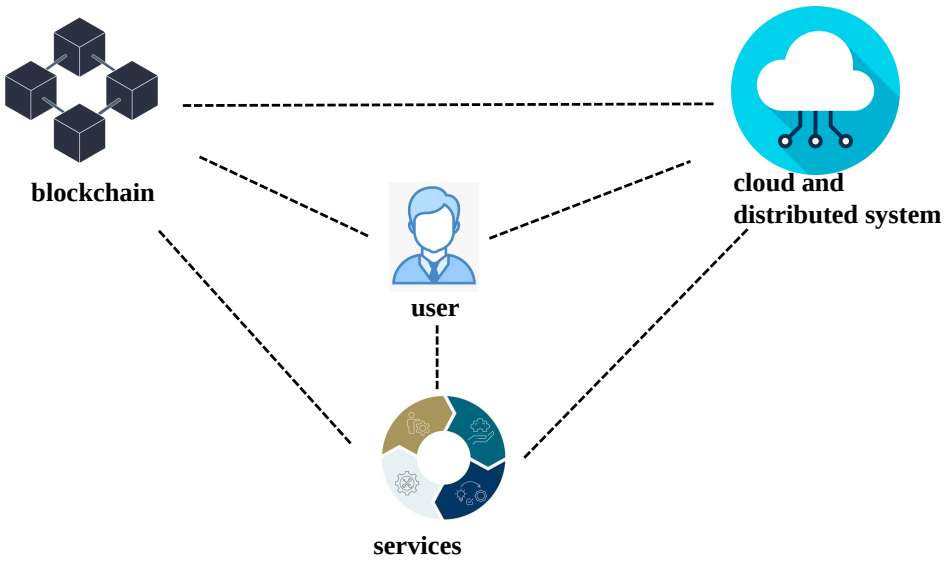

Figure 1.7: Blockchain-enabled environment platform assuring interoperability, transparency, traceability, and security

- **Real estate:** Using the blockchain mechanism to record real estate transactions can provide a more secure, trusting, and accessible way to verify and transfer ownership. This will reduce paperwork and speed up transactions to save money.
- **Voting:** If personal identity information is stored on a blockchain, this also allows us to vote using blockchain technology. Blockchain mining can ensure that no one votes twice, that only eligible voters can vote, and of course, that votes cannot be tampered with. Moreover, it can make voting easier by making it as simple as pressing a few buttons on your smartphone app. At the same time, it significantly decreases the cost of organizing an election.
- **Government benefits:** Another way to exploit digital identities stored on a blockchain is to administer government benefits such as welfare programs, social security, and health insurance. The exploitation of blockchain technology will reduce fraud and costs.
- **Healthcare and medical information:** Storing medical data on a blockchain can allow doctors and healthcare professionals to track and obtain accurate and up-to-date information about their clients (patients). This can ensure that patients who see several doctors, for example, will receive the best care possible. Also, it will speed up the medical data handling system, allowing faster processing in some cases. And, if insurance data is stored on the blockchain, doctors can easily check whether a patient is insured or not and whether or not their treatment is covered.

- **Artist royalties and Non-fungible tokens:** Using the blockchain to track music and movie files streamed over the internet can ensure artists get paid for their work. Since blockchain technology was developed to ensure that the same file does not exist in multiple places, it can help reduce hacking.
- **Logistics and supply chain tracking:** Using blockchain to track items as they move through a logistics or supply chain network will ensure several benefits. First, it facilitates communication between partners because the information is available on a public and secure register. Second, it provides data security and integrity as the data on the blockchain cannot be altered.
- **Secure Internet of Things networks:** The Internet of Things (IoT) facilitates and develops our lives, allowing malicious actors to access our private lives or take control of critical systems. Blockchain technology will ensure a high level of security by storing passwords and other information on a decentralized network instead of a centralized server.

1.7.3 Characteristics

We are going to see four characteristics of the blockchain.

- **Powerful:** Blockchain provides us with many advantages, including security, stability, transparency, distribution, anonymity, confidentiality, efficiency, and traceability.

  e have three kinds of blockchain, public, private, and hybrid. In the first type (also called: open access or permission-less), each user that has a valid pseudonym generally an (account address) can freely join the network and performed any available network functionalities such as sending, receiving, and validating transactions according to standard rules. Thus, each node can participate in the processes of consensus, although a user's voting power is generally proportional to its control of network resources, such as wealth token, computing power, and storage space [95].

  The private kind (also called permissioned or authorized blockchain) works in a restrictive setting, i.e., a closed environment. In this type of blockchain under an entity's management, only authorized users with a demonstrated identity are allowed to promote basic functionalities such as data propagation and consensus participation [96].

  A hybrid kind (also called consortium or federated blockchain) is an innovative strategy for solving the requirements of communities that have a need for private and public blockchain use. Some parts of associations are made

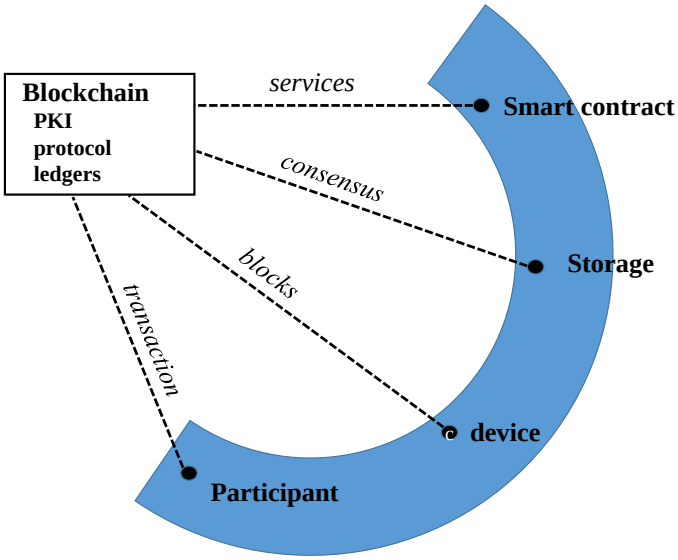

Figure 1.8: Blockchain-based environment architecture

public, and others stay private. The consensus mechanism in this type is controlled by the predefined users. This kind is not open to the popular groups, it still has a decentralized nature where there is not a single centralized identity that controls it. Thus, it presents all the functionalities of a private blockchain such as transparency, efficiency, confidentiality, etc.

- **Implementation Levels:** In the part of system design, blockchain can have four different implementation levels. These levels are protocols of data and network, protocols of distributed consensus, the framework of autonomous organization that uses smart contracts, and an interface (an application) [97].
- **Components:** To constitute a blockchain system, we have to introduce five components: private and public keys, signed chains, a big community, a protocol of peer-to-peer document distribution, and a consensus mechanism (Figure 1.8). The most important is the consensus algorithm because it is responsible to achieve a general accord to merge a new block into the blockchain.

1.7.4 Attacks on a consensus protocol

The use of an improperly functioning consensus algorithm will inevitably make the entire system unusable for reliable use and thus endanger all sensitive data. The weak consensus approach used exposes the system to a number of notorious attacks, including:

- **51% attack:** In a consensus mechanism, a group of miners working together called a mining pool can achieve domination by controlling more than half of the whole network computation (hash rate) [98]. The pool would be capable to integrate its created blocks to the blockchain or construct a competing independent fork (branch). This kind of attack permits the malicious pool to perform a double-spending attack i.e., to spend twice its own funds and reject transactions that it does not desire to be contained in the ledger.
- **Sybil attack:** One malicious user can perform a Sybil attack by creating an enormous number of identities and then employing them to exert disproportionate power; then, cheat the system to damage its trust mechanism [99].
- **DDoS attack:** This kind of attack desires to disrupt the ordinary functioning of the network by attacking nodes by sending information or to reduce the desired win outlook of another competing mining groupe [100].

1.8 Conclusion

The objective of this chapter was to provide an overview of some of the main concepts introduced for the general purpose of this manuscript. We started with the presentation of mathematical tools, definitions, and notations, some mathematical problems related to the proposed solutions were also mentioned. Next, we discussed encryption and scheme types in terms of symmetry. In the following element, the four types of a cryptosystem have been defined. After that, we talked about an important element in securing distributed systems, which is "the key exchange protocol". Then, we showed how the cloud offers a wide choice of on-demand and pay-as-you-go computing services to users according to their needs. These services come in the form of software, platform, or infrastructure and are deployed under four possible models, which are: the private cloud, the public cloud, the community cloud, and the hybrid cloud. Thanks to the advantages offered by the cloud, end users find this technology to be a good choice for services.

Despite all these solutions produced by the cloud, there is still the problem of security, more specifically privacy. This insufficiency inspires interest in our future research through the introduction of homomorphic and blockchain technology. The following chapters deal with the problem of security. In particular, computing on encrypted data stored in the cloud.

Chapter 2

Homomorphic encryption for cloud computing environment

In order to achieve the sharing of encrypted data with a third party without allowing him to access the content of this data, two techniques have been proposed. A fully homomorphic encryption based on number fragmentation, published in Expert systems journal. A partially homomorphic encryption based on separate processing of digits, published in ICFNDS conf.

Therefore, this chapter consists of the articles: "A Fully Homomorphic Encryption based on Magic Number Fragmentation and El-Gamal Encryption: Smart Healthcare Use Case, Experts systems journal, 2021." and "A Homomorphic Digit Fragmentation Encryption Scheme Based on the Polynomial Reconstruction Problem, ICFNDS '20: The 4th International Conference on Future Networks and Distributed Systems - St. Petersburg Russian Federation, November 2020.".

2.1 Introduction

Effective encryption techniques ensure the security and integrity of sensitive data, which in turn will make cloud entities operate autonomously and securely. In information technology, the world has become forced to work in the cloud. While the problem of maintaining confidentiality that appeared with the huge amount of data poses a great challenge to researchers, as their concerns about the privacy of digital data have increased, and it was necessary to focus on algorithms more broadly in parallel with the growth of communication devices Network and the significant increase in computing capabilities. However, exposure to various types of attacks remains a possibility. The adversary aims to tamper, destroy or steal sensitive data.

To keep safe sensitive data against these adversaries, traditional encryption schemes use a shared public key between the users involved in communication to encapsulate their exchanged data [36]. In several cryptosystems, the users, even clients or providers of service, have the best rights over data due to the encryption keys, mainly in favored cloud services, where the confidentiality of sensitive data is in danger. Nowadays, privacy preservation has led scientists to get on with the challenges of homomorphic schemes; in which, a third party could use the encrypted data without seeing its content.

Homomorphic Encryption (HE) is used to avoid these problems by enabling any other third entity to perform computational operations on encrypted data without the need to decrypt them in advance. The main idea of the HE based on the question, "Is there an encryption technique (Enc) whee $Enc(m_1 + m_2)$ is facile to compute using $Enc(m_1)$ and $Enc(m_2)$?" Principally, the question needs to explore if any algebraic homomorphic function can be designed. Abstract algebra is structure-preserving for two other structures of the same kind (such as two rings, two groups, two vector spaces, etc.) [101].

By taking advantage of these features, we aspire to increase the level of security in the cloud by making it autonomous in the calculation, and that is by suggesting new homomorphic encryption mechanisms.

2.2 Related works

The emergence of asymmetric schemes in 1978 led to the homomorphic cryptosystem story, and the concept of HE was discussed for the first time in RSA [32] by Rivest et al. The RSA cryptosystem take into account only multiplication operations; wich called a Partial Homomorphic Encryption (PHE).

$$Dec(Enc(m_1) \times Enc(m_2)) = m_1 \times m_2 \tag{2.1}$$

GM's technique [102] is another example of PHE. The authors proposed the first encryption scheme that is considered a probabilistic public key, which performs an additive operation that uses the following equation:

$$m_1 + m_2 = Dec(Enc(m_1) + Enc(m_2)) \tag{2.2}$$

The robustness of the GM scheme is based on the hardness of the quadratic residuosity problem (QRP). We call an integer α a quadratic residue modulo n if there is a number r such that, $r^2 \mod n = \alpha$. The QRP determines if a given number d is quadratic modulo n or not. Let $m = m_1 m_2 \dots m_i$ and $c = c_1 c_2 \dots c_i$, the GM

cryptosystem is described as follows:

$$c_i = Enc(m_i) = (y_i^2 \times x^{m_i}) \mod n, \ \forall m_i \in \{0, 1\} \tag{2.3}$$

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

The authors of [41] proposed a partial homomorphic technique from $(\mathbb{Z}_{n^2}^*, \times)$ to $(\mathbb{Z}_n, +)$. It is an additive scheme. The generated data have immense size, the unavailability of enough resources for storage and computation has led researchers to look out for the Fully Homomorphic Encryption (FHE) problem.

The authors in [103] have proposed a scheme that allows unlimited addition operations and only one multiplication which is called Somewhat HE. The authors has presented a public key homomorphic technique which uses the encryption $\psi(a_1, \ldots, a_n)$ of the variables $a_1, \ldots, a_n$ where $a_1, \ldots, a_n \in \{0, 1\}$. The presented homomorphic technique is as follows:

$$c = Enc(m) = g^m \times h^r \mod n \tag{2.4}$$

where, g, h and u are generators with ($h = u^q$), r denotes a random number $\in \{0, n-1\}$, based on a secret key p, the decryption process is $Dec(c) = log_{g'}(c')$, $c' = c^p$ and $g' = g^p$. The robustness of BGN's technique is based on the hardness of the subgroup decision problem that consists of deciding if a given number is a member of a subgroup Gp of a group G. The group G is concerned of a composite order $n = p \times q$, where p and q are two distinct primes.

Paper [104] has presented a threshold FHE technique (TFHE) using the learning with errors (LWE) assumption. The authors constructed a universal thresholdizer from the presented TFHE, which is used to treat the principal problem of single-round threshold signatures from lattices. This threshold functionality can be introduced into different systems, such as signature schemes, public-key encryption, and pseudorandom functions.

Several SWHE techniques introduce noise that prevents using an arbitrary degree of computing. In order to reduce the noise, different methods are used after each computing process. Based on ideal lattices, Gentry [51] has performed a bootstrapping. Bootstrapping allows an unlimited number of addition and multiplication homomorphic operations; this process is obtained from the minimization of noises in each computing. In [53], Van Dijk et al. presented a fully homomorphic encryption technique that is can be defined as follows:

$$c = Enc(m) = m + 2 \times r + p \times q \tag{2.5}$$

where $r << p$ and $m \in \{0, 1\}$, to recover the original message $m : m = Dec(c) = (c \mod p) \mod 2$, and the technique robustness is based on the hardness of AGCD problem (the Approximate-Greatest Common Divisor).

This technique is conceptually simpler than Gentry's scheme. The encryption technique presented uses multiplication and addition, it is applied to integers instead of based on ideal networks on a polynomial ring. Despite this, the cost involved in this simplicity leads to too large a public key size $O(\lambda^{10})$. The authors in [105] applied Brakerski's FHE scheme which is based on the learning with errors (LWE) problem for the ring-LWE (RLWE) parameter. Of course, there are limits in the worst case that were produced such as the noise caused by several homomorphic operations like bootstrapping, multiplication, relinearization. To simplify bootstrapping, the authors used a module-switching trick.

The authors used a batching method to minimize noise. There are many FHE schemes recently developed but none of these techniques demonstrate its practicability due to the massive amount of produced noise which maximizes the computational complexity [52]. In 2011, Gentry and Halevi [106] implemented an FHE system. However, this implementation result was unsatisfactory because it took a long time. To add up two 32-bit integers, the scheme takes around 900 seconds, and around 67,000 seconds to multiply two numbers. Moreover, it uses more than 780,000 bits to encrypt only one bit. The size of a public key requires from 70 Megabytes to 2.3 Gigabytes depending on the setting size. The bootstrapping process on a 1-CPU 64-bit machine with large memory, takes a time range from 30 seconds to 30 minutes depending on the setting size.

The authors in [107] presented a fully homomorphic encryption technique, the encryption process is as follow:

$$c_i = (m_i + rand_i \times k) \mod (k \times p) \tag{2.6}$$

with $rand_i = (m_i \times k) \mod p$ and $f^{-1} : m_i = c_i \mod k$. In order to perform an order-preserving system, the authors used the following linear formulation: $a \times x + b$. The number b is kept secret to mask the value of other parameters, the value of $a \times x$ will be masked by b using an addition operator. Linear formulations have to respect the indexing order. As a definition of the order-preserving function, the authors used the following equation: $index_i = p \times m_i + rand_i$.

The authors in [108] presented a new homomorphic encryption method for integer arithmetic. This technique is based on the Chinese Remainder Theorem (CRT) secret sharing. The hardness of their system relies on an assumption of the toughness of the partial approximate common divisor problem (PACDP).

The paper [109] introduced a homomorphic evaluation of the prince block cipher.

The authors implemented a generalization of NTRU. They used a decreasing sequence of primes numbers $q_0 > q_1 > \cdots > q_d$ in KeyGen process. To encrypt, they used the following formula:

$$c^{(0)} = h^{(0)} \times s + 2 \times e + b \tag{2.7}$$

s and e are random numbers. To decrypt the ciphertext c with , the decryption process consist of multiplying the encrypted text by the corresponding private key $f^{(i)}$; then, computes the original text by modulo 2 as : $m = c^{(i)} \times f^{(i)} \mod 2$.

The authors in [110] exploited El-Gamal [39] cryptosystem as an original encryption method. They used three parts to present a ciphertext, parts one and two are almost similar to El-Gamal. Parts two and three are decrypted using the first part α^k which is random. Part two equals $m \times \beta^k$ with $\alpha^p = \beta$, the sum of two numbers is calculated by: $c_3 = \beta^{k+m}$. Unlike them, we use a linear formula to compute a sum of two numbers, that is more faster to decrypt. To decrypt $\beta^{m_1+m_2}$ for recovering $m_1 + m_2$ requires a very long time.

Deterministic encryption techniques like RSA cryptosystems and Modified Paillier encrypt the same text more than once, and it will always produce the same encrypted text. Formally, if $c_1 \neq c_2 \Rightarrow m_1 \neq m_2$ with $Enc(m_1) = c_1$ and $Enc(m_2) = c_2$. Deterministic schemes do not satisfy IND-CPA (Indistinguishability under Chosen Plaintext Attack) level security, so these schemes are vulnerable to Chosen Plaintext Attack. Therefore, they are not semantically secure. In CPA, the adversary having c and he wants to get m with $c = Enc(m)$, he select m' and computes $c' = Enc(m')$. Then, the adversary decrypts $c \times c'$ to get $m \times m'$ and obtain m.

Also, deterministic encryption does not secure FTG (Find-then-guess) attack. In FTG, the adversary requests the Oracle model for two encryption, the model responses with one encryption of these two messages. The adversary has to guess what message the Oracle model encrypted. A technique is secure FTG if the adversary cannot easily distinguish which message was sent (encrypted). Therefore, a deterministic technique cannot be secure because the adversary can distinguish. In our proposed scheme, we face this kind of attack by using a probabilistic method where the same text can be encrypted by many values. Formally, there are m and m' with $m = m'$ and $c \neq c'$.

Thus, the goal is to increase the efficiency of encryption systems from several aspects, including increasing the speed of operations and reducing the volume of encrypted data, so by developing two encryption techniques that we will explain in the next point.

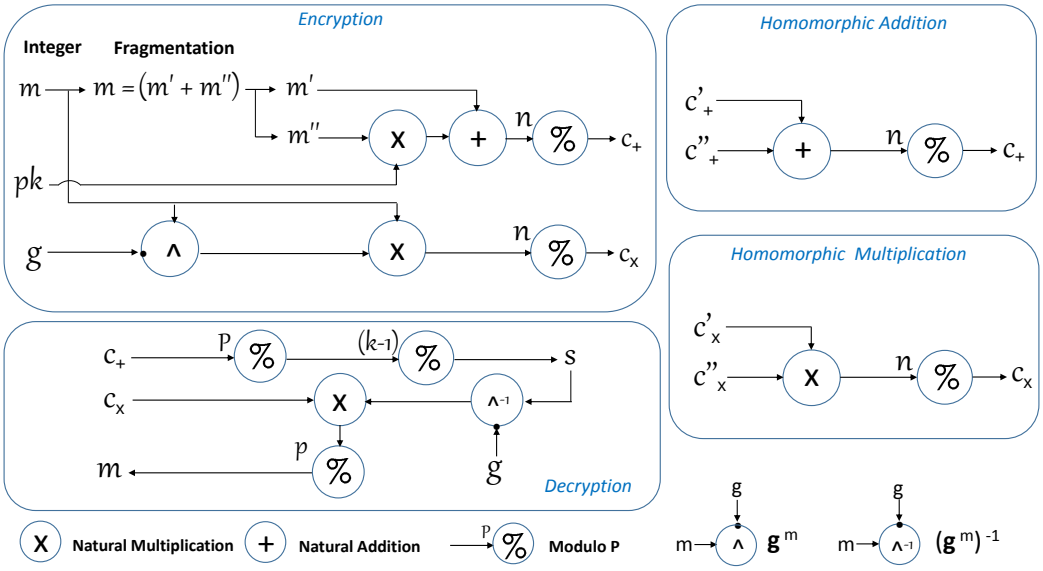

Figure 2.1: Homomorphic property

2.3 Proposed techniques

First technique

In [7], we presented an FH encryption technique in order to give a secure and robust cryptosystem with an unlimited number of holomorphic operations (addition and multiplication). The presented homomorphic scheme could be implemented in an expressive cloud architecture as demonstrated by Figure 2.1. In this scheme, the encrypted text is separated into two parts. The first one is a linear formula scheme that is used to guarantee a homomorphic addition process. On the other hand, this part is used also as a parameter for the decryption of the second one which guarantees the multiplication process.

The homomorphic addition process concentrates on number fragmentation and factorization problem, whereas the multiplication homomorphic process uses the El-Gamal cryptosystem principle with some modifications. In our scheme, the used generator g is an integer separated from public key y; in the original El-Gamal, $y = g^k$ with k is a private key. Also, we do not use a generated random number for each text to be encrypted; in original El-Gamal, $c = (m \times y^r, g^r)$ with r is a random integer. In proposed MNF-G, $c = m \times g^m$. So, the power of integer g equals m with m denoting the original text to be encrypted. We note here that m is the decrypted result of the first part. The second part is based on the discrete logarithm problem. Also, the exploitation of the El-Gamal system leads us to speed up the power computation (to compute g^m) of big numbers. The got results are indicated in Subsection 2.4.2.

RSA too could give homomorphic multiplication properties. However, El-Gamal is used here since it guarantees text integrity issue. For example, if we use RSA, we will obtain $c+c' = (m+m', (m\times m')^k)$ with $m = m_1\times k+m_2$ and $m' = m'_1\times k+m'_2$; by using El-Gamal, we will obtain $c+c' = (m+m', (m\times m')\times g^{m+m'})$. We see that in El-Gamal encryption $m+m'$ emerges on both sides addition and multiplication. Thus, the first part decryption result can be used to decrept the second one.

There are many symmetric FHE techniques that provide good efficiency regarding speeding and resource distribution. However, these techniques suffer from several critical issues. The key-sharing problem is one of the met situations since the communicants should utilize a private channel to share a symmetric secret key. Asymmetric techniques provide more hardness compared with symmetric techniques. But the complexity of computing and resource use of the asymmetric schemes is the major problem. The offered MNF-G system reduces the calculation complexity by using two ciphertext parts. However, many other asymmetric encryption schemes require a very long operation time because they use exponential computation. We used a linear asymmetric formula that accelerates the encryption process time and provides a strong and secure cryptosystem against attacks. We begin the illustration of the presented FH technique with the following linear formula:

$$c = (m + r \times p) \mod n \tag{2.8}$$

With $n = p\times q$. The weakness of the scheme defined in Equation (2.8) is vulnerable to a known plain text attack. To manage this weakness and other types of attacks, some improvements are introduced to Equation (2.8)

$$c = (m \times k + r \times p) \mod n \tag{2.9}$$

However, Equation (2.9) is still vulnerable because if an adversary can get two plain texts, he will be capable to retrieve the secret p and then decrypt all other encrypted texts. The aim of our technique is to disguise the message before utilizing Equation 2.9 by performing a random fragmentation of m into two values m' and m'' where $m = m' + m''$ so that it very hard for an adversary to find m' or m'' or to get p. Thus, the suggested encryption equation is as follows:

$$c = (m' + m'' \times k + r \times p) \mod n \tag{2.10}$$

The fundamental manners of encryption, decryption, addition, and multiplication homomorphic are displayed in Figure 2.1. To realize a homomorphic encryption property, some challenges have emerged due to the multiplication in $c_1 \times c_2$ because it will generate k^2. To guarantee homomorphic multiplication, we denote a ciphertext c into two values: c_+ and $c_\times$ where $c_\times$ is calculated by using the El-Gamal encryption.

The El-Gamal is an efficient technique that gives a high level of confident communications [111]. El-Gamal (invented by T.El-Gamal in 1985) presents an asymmetric encryption scheme with homomorphic properties which is founded on the Diffie-Hellman key exchange protocol. Its scheme relies on the one-way function, signifying that both the encryption and decryption processes are done in different functions. In this technique, the key generation process is a finite cyclic group $\mathbb{Z}_p^*$, with p denotes a large prime number, a generator g must be calculated, then a random integer α where $1 < \alpha < p-1$ is selected. The private key α have to be secret, and the public key is (g, β, p) where $\beta = g^\alpha \mod p$. To encrypt a text m, a random integer $k \in \{1, \ldots, p-1\}$ is selected for each text, and $Enc(m) = (X, Y)$ where $X = g^k \mod p$ and $Y = (m \times \beta^k) \mod p$. The couple (X, Y) will be decrypted as $m = (Y \times ((X)^\alpha)^{-1}) \mod p$.

In this point, the security is founded on the complexity of cracking the discrete logarithm problem with the modulo of a large prime. In [112], the authors mention that if the message space is not large, then m_1 and m_2 could rather be encrypted as g^{m_1} and g^{m_2} , respectively; in this manner, $(X_1 \times X_2, Y_1 \times Y_2)$ could also be regarded as an encryption of $(m_1 + m_2) \mod p$ encapseled by $g^{m_1+m_2}$. Many variants of the El-Gamal scheme have emerged, such as the original El-Gamal encryption technique, the El-Gamal system in Z_n with n is not necessarily to be prime, the El-Gamal scheme in the field of Gaussian integers, the El-Gamal scheme over quotient rings of polynomials and the bilinear El-Gamal encryption technique that is utilized in a secure privacy-preserving data collection [113].

The proposal design
In this point, the proposed MNF-G (Magic Number Fragmentation and El-Gamal encryption) will be detailed. The 'Magic' notation here means nothing except random fragmentation to increase the level of security by increasing the number of possibilities for an opponent to try.

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

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

There are three processes in the presented fully homomorphic encryption technique as follows :

- KeyGen: that gives both secret and public key where $(sk, pk) = (k, k + r \times p, g)$
- $Enc(m)$: a message $m \in P$ is encrypted utilizing the public key, $c_+ = (m' + m'' \times pk) \mod n$, and $c_\times = m \times g^m) \mod n$
- $Dec(c)$: an encrypted message is decrypted utilizing the secret key, $m = (c_+ \mod p) \mod (k-1)$, and $m = c_\times \times (g^m)^{-1} \mod p$

The robustness of MNF-G scheme is based on the hardness of number fragmentation and factorization problem. Table 2.1 indicates the data stream between the cloud, data owner, and users, with $m < k - 1$.

Time	Type	User/owner	Cloud
t_0	Secret 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	Homomorph 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 2.1: The data stream between the cloud, data owner, and users.

MNF-G example
Let:
$p = 207472224677348520782169522210760858748099647472111729275299258991219668475054965831008441673255007 7$

$q = 1814159566819970307982681716822107016038920170504391457462563485198126916735167260215619523429714031$

$k = 307625422503012706920514605395861669272917327549 61$

$g = 607538052934545886014457739870476161464 9$

$r = 5514159566819970307982681716822107016038920170504391457462563485198126916735167260215619523429714031$

$m = 122305478$

We obtain

$c_+ = 947952240942614920035319208414098289421918610716727151898985906593176253026927376399799518829105424568128853058439450804689218249315376014213447740207108127265026121211249830050091350639267856294740$

$c_\times = 154319883806791958384076633855737991429837550709947179579371636838668051441989419762945987159138056072958644433567409647697660267629789272086120690048112199932580966879148170109053424361832764983406$

Second scheme
In [12], a LFHE scheme is proposed offering a high level of security with the carefulness of hardness. Thus, Equation 2.11 is used to encrypt data homomorphically:

$$\psi = (\pi + r \times p) \mod n \tag{2.11}$$

with $n = p \times q$, p and q are two big prime numbers, r is a random integer. This encryption realizes fully homomorphic properties where $\forall \pi < p$. Unfortunately, an adversary can easily find the secret p then decrypt all cipher data. To enhance this encryption technique, we treated each digit of a message m separately rather of treating the entire m. In another words, each digit of text that denoted m_i is from 0 to 9 so $m_i \times k < m \times k'$ with $k < k'$. Therefore, we assume that m is written in a base k and it will be converted to base 10. To recover the original text, we successively divide the encrypted message c by k^i.

Digit Fragmentation Encryption Scheme (DFE) consists of these operations:

- KeyGen: the process that gives both secret and public key, with $(sk, pk) = ((k, p), (k + r \times p, n))$
- $Enc(m)$: a message m is encrypted by employing the public key, $c = m_0 \times pk^0 + m_1 \times pk^1 + \ldots m_i \times pk^i$
- $Dec(c)$: an encrypted message is decrypted by employing the secret key, $m = \sum_{i=l}^{0} (\frac{c}{k^i}) \times 10^i, c \leftarrow c - (\frac{c}{k^i}) \times k^i$

This technique concentrates on the hardness of both the polynomial reconstruction problem the number and factorization problem where $(\frac{c}{k^i})$ is the quotient of $c \div k^i$. In deciphering, one must start by dividing over k with the greatest power in order to extract the last digit, by which it is impossible to extract the first digit first; if we currency c on the lowest power, we cannot recover the first digit.

DFE example
Let:

$p = 721261014729547490954452378504349240996938214818676546008250008539351955652592145558870542302075142 1$

$q = 976522637021306403150551933319006137720124048624544172072735055780411834104862667155922841$

$k = 5754853343$

$r = 92040057282660900487772532072414166690514763692165012667548138216199844722247808764883442 79$

$m = 122305478$

We obtain

$c = 67692075165066868414340918672250760286466292844385914931870794295354808013674841315768396707703712011059386182465558229859074613278463282450050109437020376291242822959954064152073943778168 27$

2.4 Performances

In this Section, we demonstrate the characteristics of the proposed schemes.

2.4.1 MNF-G

Encryption
An asymmetric encryption scheme allows Alice to send Bob an encrypted text encapsulated by the shared public key that operates as described in Figure 2.2. where $pk = k + r \times p$ and of $c = (m' + m'' \times pk) \mod n$, that means, $c = m' + m'' \times (k + r \times p) = m' + m'' \times k + r^{'} \times p$. $m' < p \rightarrow c \mod p = m' + (m'' \times k) \mod p$ and $m'' < p \rightarrow (m'' \times k \times k^{-1}) \mod p = m''$

Decryption
We have $c = (c_+, c_\times)$ with $c_+ = m' + m'' \times k + r \times p$. Firstly, we recover m from c_+ where $m \times k < p$. So, $c_+ \mod p = m' + m'' \times k$ with $r \times p \mod p = 0$. $m \times k \mod (k-1) = m$ so $Dec(c_+) = m$. Then, this result will be used to decrypt the second encrypted value $c_\times$ by introducing g. Let $s = Dec(c_+) = m$ we obtain $m = (c_\times \times (g^s)^{-1}) \mod p$.

Homomorphic Addition characteristic
We have $Enc(m_1) + Enc(m_2) = m'_1 + m''_1 \times pk + m'_2 + m''_2 \times pk = (m'_1 + m'_2) + (m''_1 + m''_2) \times pk = E(m_1 + m_2)$.

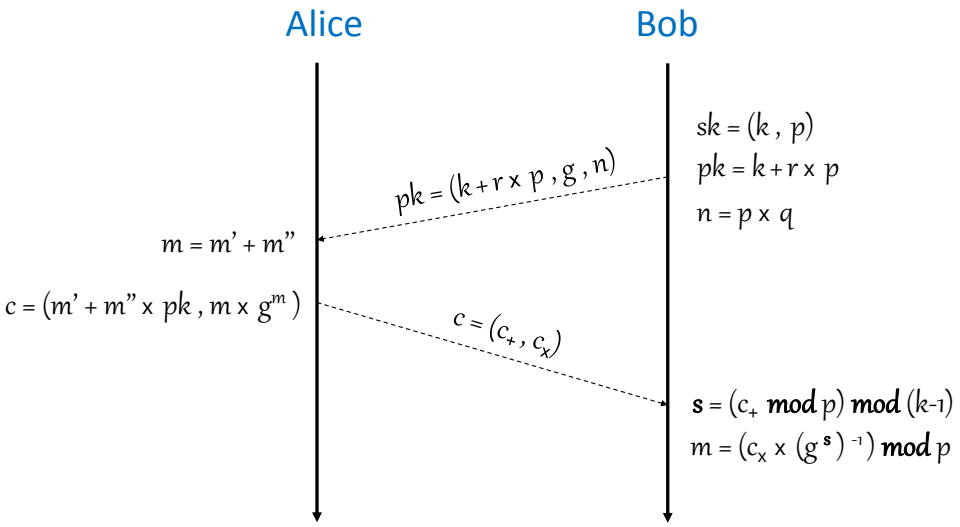

Figure 2.2: The encryption technique

thus

$c = c_1 + c_2 + \cdots + c_i = (m'_1 + m''_1 \times pk) + (m'_2 + m''_2 \times pk) + \cdots + (m'_i + m''_i \times pk)$ $= (m'_1 + m'_2 + \cdots + m'_i) + (m''_1 + m''_2 + \cdots + m''_i) \times pk = m' + m'' \times pk$, with $m' + m'' < p$.

We obtain $Dec(c) = m_1 + m_2 + \cdots + m_i = m$. Generally:

$$\sum_{i=1}^{t} Enc(m_i) = Enc(\sum_{i=1}^{t} m_i) \tag{2.12}$$

Homomorphic multiplication characteristic
Let respectively c_1 and c_2 be two encrypted messages from $m_1 = m'_1 + m''_1$ and $m_2 = m'_2 + m''_2$.

$Enc(m_1) \times Enc(m_2) = (m_1 \times g^{m_1} \times m_2 \times g^{m_2}) = (m_1 \times m_2 \times g^{m_1} \times g^{m_2}) = (m_1 \times m_2 \times g^{m_1+m_2}) = (m \times g^s)$ with $s = m_1 + m_2$.

Remarking that $s = Dec(c_1 + c_2)$, in order to ensure the multiplication characteristic, it is required to ensure the decryption of $(m \times g^s)$, this process uses the addition value of the first part.

We have $Enc(m_1) \times Enc(m_2) = (c_{1+} + c_{2+}, m_1 \times m_2 \times g^{m_1+m_2})$. Generally:

$$\prod_{i=1}^{t} Enc(m_i) = Enc(\prod_{i=1}^{t} m_i) \tag{2.13}$$

2.4.2 An ameliorated power method

The disadvantage of the exponential calculation is a long time of computing; particularly with low-capacity instruments. To make our technique faster, we used an ameliorated exponential computation manner in order to overcome this difficulty. The vision is to transform the power calculation into a multiplication operation, by computing the vector vec using the input size with consideration to $\mod n$, which we designate by $len(m)$ for an input m, then we compute g^m by employing this vector. More exactly, let $m = m_0 m_1 m_2 \dots m_i = m_i \times 10^0 + m_{i-1} \times 10^1 + \dots + m_0 \times 10^i \Rightarrow g^m = (g^{10^0})^{m_i} \times (g^{10^1})^{m_{i-1}} \dots \times (g^{10^i})^{m_0}$. so, we just have to compute $vec = [g^{10^0} \mod n, g^{10^1} \mod n, \dots, g^{10^{n-1}} \mod n]$ then we get the following result:

$$g^m = \prod_{i=0}^{t} vec_i^{m_i},\ with\ 0\ <=\ m_i\ <=\ 9\ and\ t\ =\ len(m) \tag{2.14}$$

Table 2.2 compares between used $vec-power$ method and pow the power function of python library, utilizing an HP Laptop with Intel(R) Core(TM) i3 and RAM of 4 Go. Test 'one': $len(m) = len(g) = len(n) = 1400$ digits. Test 'two': $len(m) = len(g) = len(n) = 2100$ digits.

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

Table 2.2: The ameliorated power method tests

Concerning the improvement of the complexity in this method, if for example $v = g^{17}$, and $g^{27} = v \times g^{10}$, then our method $g^{27} = v \times vec_1$, since $g^{17} = vec_0^7 \times vec_1^1$ and $g^{27} = vec_0^7 \times vec_1^2$. If logarithmic time complexities generally apply to algorithms that halve problems each time, and the complexity of Python's power function is known to be $O(logn)$, then the complexity of our method is $O(log(logn))$.

In the next point, we will demonstrate some performances and characteristics of our second encryption scheme entitled "A Homomorphic Digit Fragmentation Encryption Scheme Based on the Polynomial Reconstruction Problem (DFE-PR)".

2.4.3 DFE-PR

Encryption
Let $m = m_i m_{i-1} \dots m_0$ with m_0 denotes the weak part of m, $m_j \in \{0, .., 9\}\ \forall\ j \in$

$\{0, .., i\}$
$c = m_0 \times pk^0 + m_1 \times pk^1 + \ldots m_i \times pk^i = m_0 \times k^0 + m_1 \times k^1 + \ldots m_i \times k^i + r \times p.$

With consideration of: $m_0 \times k^0 + m_1 \times k^1 + \ldots m_i \times k^i < p.$

Algorithm 1 demonstrates the encryption process.

Algorithm 1 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 1. $(k-1) \times (k^0 + k^1 + \ldots k^{i-1}) < k^i$

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

If $m_i <= 9 \wedge 9 < k - 1 \Rightarrow 9 \times k^0 + 9 \times k^1 + \ldots 9 \times k^{i-1} < k^i \Rightarrow \frac{c}{k^i} = m_i$

so $c \leftarrow c - (\frac{c}{k^i}) \times k^i$ then, we will get

$m_0, m_1, \ldots m_i$, by computing $c = m_0 \times 10^0 + m_1 \times 10^1 + \ldots m_i \times 10^i$ and to guarantee the original message recovery, we use the division on k^i with i denotes the length by digit of cipher c, put $i > j$ if j is the length by digit of a message m due to $k > 10$ ($\Rightarrow c > m$). Similarly, the decryption operation is correct for $m_i < k$ (Equation 1). Algorithm 2 demonstrates the decryption process.

Algorithm 2 Decryption algorithm

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

```
1: function DEC
2:     c ← c mod p
3:     l ← length(c)
4:     m ← 0
5:     for i ← 0 to l + 1 do
6:         d ← c / k^(l−i)
7:         c ← c − d × k^(l−i)
8:         m ← m + d × 10^(l−i)
9:     end for
10:    return m
11: end function
```

Homomorphic Addition characteristic
Let $c_1 = m_{01} \times k^0 + m_{11} \times k^1 + \ldots m_{i1} \times k^i + r_1 \times p$,

and $c_2 = m_{02} \times k^0 + m_{12} \times k^1 + \ldots m_{i2} \times k^j + r_2 \times p$ this implies

$Enc(m_1) + Enc(m_2) = (m_{01} + m_{02}) \times k^0 + (m_{11} + m_{12}) \times k^1 + \ldots (m_{1i} + m_{1i}) \times k^i$ so

$Enc(m_1) + Enc(m_2) = m'_0 \times k^0 + m'_1 \times k^1 + \ldots m'_i \times k^i + r' \times p.$

with $m'_x \in \{0, .., k-1\}\ \forall\ x \in \{0, .., l\}$, thus $Enc(m_1) + Enc(m_2) = Enc(m_1 + m_2)$.

2.5 Techniques analysis

Cryptanalysis can be involved in multiple types of known attack scenarios. In asymmetric encryption systems, the public key is known to everyone; if the public key is not selected carefully, it can make this kind of encryption weaker than symmetric systems and this can guide some critical data.

2.5.1 MNF-G analysis

MNF-G security analysis
We will talk about analyzing the hardness of MNF-G. The enemy's purpose is either to get secret data or to know some secret used primitives. In [7], we used

two values c_+ and $c_\times$ to determine a ciphertext. We begin with c_+ and the possible attacks. The proposed MNF-G is concentrated on the factorization problem. From a couple of the MNF-G public keys, and when the trapdoor is not well-locked, some hypothetical mathematical characteristics can be manipulated by different cryptanalytic attacks to disclose the private key. Some known attacks are based on weaknesses of factorization with small prime numbers.

We put a plain text m encrypted by $pk = (k + r \times p)$ so $c = m' + m'' \times pk$, its encryption by another public key $pk' = (k + r' \times p)$ is $c' = m' + m'' \times pk'$. Thus, $(c - c')$ denotes an encryption of value 0 that equals $r'' \times p$. This issue conducts to disclosing the secret value p. Therefore, in our MNF-G, the public key is computed in a unique manner in order to avoid the 0 encryption case.

Figure (2.3) indicates the plaintext fragmentation impact on the time in order to discover a solution using a brute-force attack, the attaker commences trying all possible numbers of a private key from biggest to smallest or vice versa. In an asymmetric technique over integers such as RSA when $c = m^d \mod n$ and $m = c^e \mod n$, the public key d is known, the attacker attempts to compute $m = c^e$ for any value of m selected. To get the private key e, the attacker has to use all probable values in range $\{1, e\}$. The same case with the encryption techniques such as original El-Gamal [39], Paillier [41], and BGN [114]. Considering that t denotes the time to execute an operation and x is the private key size. The formula that defines the required time for a successful brute-force attack is $f(x) = x \times t$.

The presented strategy is based on the message fragmentation by which the enemy has to try all the possible values of the private key for each probable fragmentation; so, for fragmentation 1, $c = 1 \times k + (m - 1)$, the enemy should test all the values of k. For fragmentation 2, $c = 2 \times k + (m - 2)$ the adversary enemy test all the values of k, etc. Therefore, it has $\frac{m}{2} \times x$ trials. Considering the size of m equals x, it will take $f(x) = \frac{x}{2} \times x \times t$. Finally, we can formulate:

$$f(x) = \frac{x^2}{2} \;\; in\ MNF - G\ and \;\; f(x) = x \;\; in\ other\ schemes \tag{2.15}$$

In MNF-G, the encryption method is more similar to the $m \times k$ method. This encryption example is clearly vulnerable to some attacks. Using a public key (n, pk), the length of pk will give the number of probable combinations of $(k_i, r_i \times p_i)$ where $pk = k_i + r_i \times p_i$. An adversary can commence recovering $r \times p$ or k employing all potential combinations. This point conducts to the number fragmentation and factorization problem. The adversary has to try with all potential combinations of k, and then factorize the remainder of $pk - k_i$, which is equivalent to testing a factorization of $(pk - 2)$ numbers.

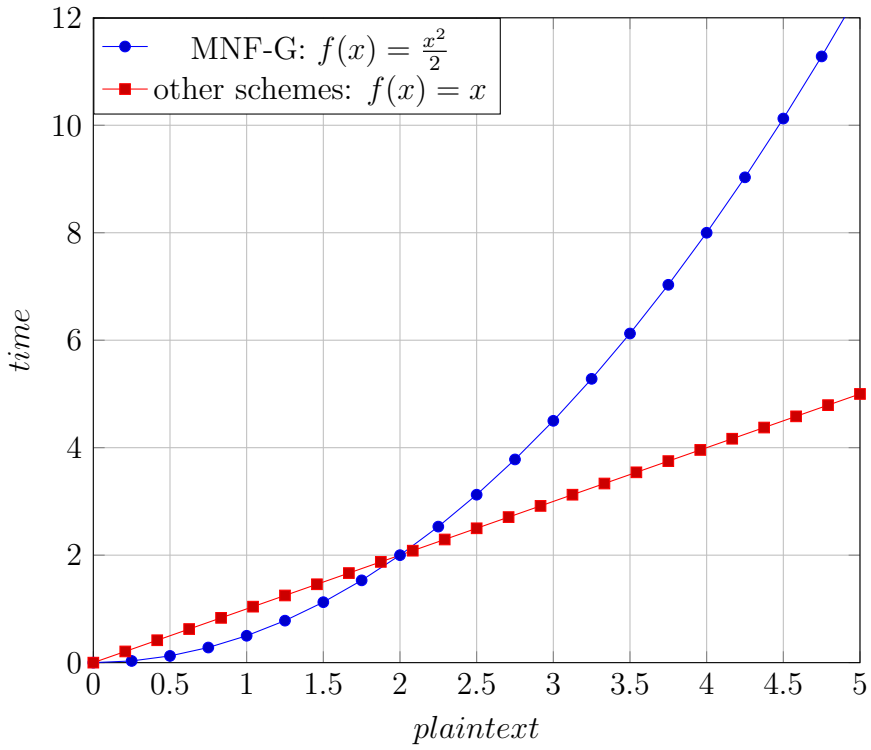

Figure 2.3: Time comparison for a successful brute-force attack

If pk is large, it is very hard to perform all these factorizations currently. The first value in MNF-G is a probabilistic technique showing a hard requirement to satisfy IND-CPA (Indistinguishability under Chosen Plaintext Attack). In fact, $c_\times$ is deterministic, that is not a disadvantage in our system. In CPA, the adversary choose a new m' and compute $c' = Enc(m')$; the he decrypts $c \times c'$ utilizing Oracle request in order to get $m \times m'$ and discover m, then k. This sort of attack is practical for some schemes such as RSA, because $c \times c' = (m \times m')^e$, in another word, $c \times c'$ only includes $m \times m'$ and the private key. But in MNF-G scheme, $c \times c' = m \times m' \times g^{m+m'}$, which makes it more hard to decrypt it in order to get $m \times m'$.

The second element $c_\times$ in MNF-G is a modified El-Gamal scheme; in which, the power of the generator g is not a random number in each plain text like the original El-Gamal, but it represents the message m itself, and this does not change the proposal hardness which consistently relies on the discrete logarithm problem (DLP). The principal attack on the El-Gamal scheme is to crack DLP: given both α and α^x from the finite field $\mathbb{Z}_p$, then discover the value of x. To construct a hard DLP, it should choose a prime number p with $p-1$ having a big prime factor. Nevertheless, if $p-1$ is only dividable by prime integers less than positive number β, the DLP can be reduced to locating a small number of size β rather than the size of $p-1$.

The DLP solution can be found for $log_\alpha(\beta)$ by computing $1, \alpha, \alpha^2, \ldots$ until detect a match with β, then α could be changed by $\alpha_1 = \alpha^m$ and β by $\beta_1 = \beta^m$ to solve $log_{\alpha_1}(\beta_1) \mod r$ where $r \times m = p - 1$. Thus, it should construct a list $1, \alpha_1, \alpha_1^2, \ldots, \alpha_1^x$ of size at most r before finding β_1.

The original El-Gamal scheme is vulnerable to the man-in-the-middle attack, when an eavesdropping Eve is placed between two users Alice and Bob to get their data. Considering to the next example, Alice picks a secret key x, constructs a public key (g, y_1) with $y_1 = g^{x_1}$, and transmits it to Bob; this data is intercepted by Eve. The attacker selects a private random number z and constructs another public key (g, y') with $y' = g^z$ and she transmits it to Bob, acting to be Alice. At the same time, Eve transmits this duplicate key (g, y') to Alice by acting as Bob. The attacker Eve has establish a common session key equals $g^{x_1 \times z}$ Alice and anther common session key equals $g^{x_2 \times z}$ with Bob, mentioning that Bob public key is (g, y_2) with $y_2 = g^{x_2}$). Now, Eve can get exchanged data between Bob and Alice, also she can decrypt, re-encrypt, and resent it. To bypass this kind of attack, we have to use a method where Alice would not have confounded Eve with Bob.

The original El-Gamal in $\mathbb{Z}_p^*$ demonstrated that it is protected against Chosen-Plaintext Attack (CPA) despite not being against Chosen-Ciphertext Attack (CCA) [115]. In fact, for a strategy to be secure against CPA, the Decisional Diffie-Hellman hypothesis must be respected. This level requires many settings, such as secret key size, the secret p must be a safe prime number i.e., of the form $2 \times q + 1$ with q is also prime, and the selection of a generator g which denotes a primitive root. Regarding CCA for MNF-G, a technique is not protected against CCA if the encryption of a multiplication is the multiplication of the encryption, e.g., $Enc(m_1 \times m_2) = Enc(m_1) \times Enc(m_2)$. For example, RSA and El-Gamal cryptosystems. In MNF-G, we have $Enc(m_1 \times m_2) = (m_1 \times m_2) \times g^{m_1 \times m_2}$

and $Enc(m_1) \times Enc(m_2) = (m_1 \times m_2) \times g^{m_1 + m_2}$. Thus, the proposed MNF-G is safe against CCA.

MNF-G complexity analysis
The presented MNF-G technique is defined in two parts. In the encryption of the first one, there are three processes, fragmentation, product, and sum. It present a complexity of $C(n+1) = C(n)$, so the calculation complexity of the first procedure is $O(1)$ in the first part and $O(log(logn))$ in the second part. In the decryption process of the second value, we compute g^m where its complexity equals $O(logn)$. This result is improved and reduced calculation complexity compared with other techniques such as [116] where its complexity is $O(n^4)$ and the complexity of [117] is $O(n^{13})$; in addition, we can use our ameliorated method to reduce this result.

The first encrypted value can be decrypted using two operations $((c \mod p) \mod (k-1))$, that provides a calculation complexity equals $O(1)$. To decrypt the second value, we have to compute $(g^m)^{-1}$ that provides a complexity equals

$O(log(logn))$. Compared with [116] of a decryption complexity equals $O(n^4)$ or with [117] of a decryption complexity equals $O(n^{12})$, proposed MNF-G give a considerable calculation complexity.

2.5.2 DFE-P analysis

DFE-P security analysis
DFE-PR depends on two classes of hardness, the first is to get the private key p, i.e., resolving the factorization problem which consists in determining prime factors of a given number. Until today, there is no known manner to get p for big size of $n = p \times q$. Scientists confirm that there is no algorithm executing in polynomial time that finds the prime factors of any given number. Factoring a given number of 200 digits (670 bits) demands 4 billion years of computation where its time complexity is sub-exponential $2^{O(n)}$. The best strategy for the factorization of numbers have a complexity $O = 2^{n^{1/3}}$. Currently, the minimum size of a public n must be equal to 2024 bits, for use after 2022, the minimum size of n must be equal to 4096 bits.

To get the private key k, the second robustness class of our technique is founded on the Polynomial Reconstruction Problem. It relies to polynomial ring homomorphisms $R[x] \mapsto R[x]$ which could be formulated as $c(x) = r(k(x))$ with $k(x)$ denotes a key. Its general formula is $c(x) = r(x) \times k(x) + m$ such as [118] system. Paper [119], discussed the safety of this symmetric technique against a ciphertext-only attack (COA), a known-plaintext attack (KPA), and a brute force attack (BFA) utilizing two methods. The first one operates the univariate polynomial decomposition strategy and handles in polynomial time.

The second strategy uses the computation of polynomial factorization and polynomial GCDs, its time complexity relies polynomially on the encrypted text degree and logarithmically on the public n size. They found that it is very weak against a KPA. For COA, they discovered that the probability of giving a correct private key is high enough, and the BFA gets a correct key with a probability $\rho \approx 1$. But this study considered that the scheme is symmetrical, in another word, encryption without noise ($r \times p$). Introducing a noise will change the equilibrium so that the use of $r \times p$ changes $k(x)$. Thus these three attacks are inefficient, and so only factorization solving can be used by an attacker.

DFE-P complexity analysis
While the message is considered as a set of digits, numbers n and $n + 1$ contain generally the same size (the same number of digits), therefore the same computing cost. So, we can say that $C(n + 1) = C(n)$; which give a calculation complexity equals $O(1)$. Decryption process relies on the size of encrypted text, so $C(n+1) = C(n) + 1$; that offers a computational complexity equals $O(n)$.

2.6 Implementation

To estimate the presented cryptosystem ([7]) and to demonstrate that it is more efficient compared with other schemes, we performed three experimentations, results are shown in Table 2.3. Also, the result indicated in Table 2.2 denotes another improvement in our technique performance.

Table 2.3 shows the results of the comparison of four techniques performed by three tests for each one. MNF-G tests have been used in an HP Laptop with these characteristics: Processor Intel(R) Core(TM) i3-3110M CPU @ 2.40 GHz, 2 Core(s), and 4 Go RAM. TFHE-DDA is the implementation of [110] technique. The Yah-Scheme is the implementation of [120] scheme where it is performed in a PC with bi-cores Intel Core i5 CPU running at 2.40 GHz, 512KB L2 cache, and 4GB of RAM. The fourth line is dedicated to the Than-Scheme of [121], the implementation of this system has been performed in a private cloud using Eucalyptus operating on a 2.50 GHz Intel Core i5-3120 M Processor and 16 GB RAM.

We have implemented [110] scheme in order to be capable to give a comparison. About the time of keygen function, the results were comparative because we have similar variables (number/size). The cause of the distinction in time execution is that [110] technique has an supplementary calculation in $\beta = \alpha^p$.

There are three processes, *Enc*, *Dec*, and *SumPrd* that ensure the sum and product at the same time. For *Enc*, the got results display that our HE time execution is less compared with [110] scheme in all examinations. This is rational due to the number of processes. Also, [110] encryption is performed in three parts.

Decryption is also an important process because it is performed by the user, who usually has only low resource capacity. The entire time of decryption in MNF-G (sum/product) is almost similar to the decryption duration [110] second part. The primary distinction between MNF-G and [110] scheme is in the decryption of its third part where $c_3 = \beta^{m_1+m_2}$ that requires using discrete logarithm to calculate the result of a sum. We have implemented three algorithms in order to decrypt β^m which are Pollard rho, Pohling Hellman, and Baby-step giant step, but none of these algorithms can give a result within a practical duration compared with MNF-G. About size, MNF-G utilizes two parts that define the encryption text of order $2 \times n$, while [110] scheme utilizes three parts of order $3 \times n$.

The authors of [122] utilized a desktop computer of an Intel Core2 Duo E8500 CPU at 3.12 GHz. They executed their technique with a secret key size $\eta = 2176$ bits where MNF-G utilizes $p \approx 700$ digits ≈ 2300 bits, the *Enc* execution time equals 10 s vs. 13.7 ms in MNF-G. The *Dec* execution time equals 0.02 s vs. 19.4 ms in MNF-G where the public key size equals 89 MB vs. 7.5 KB in MNF-G.

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 2.3: Processing time comparison of the proposed MNF-G scheme with other works

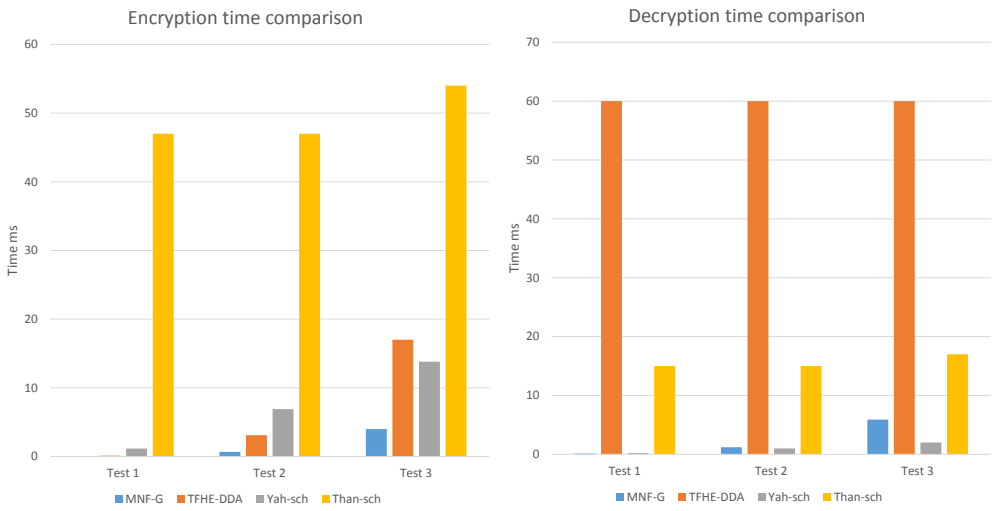

Figure 2.4: Encryption Decryption time comparison

Table 2.3 demonstrates MNF-G efficiency compared with other techniques. This efficiency is conducted by decreasing complexity and bypassing the usage of more processes like relinearization, bootstrapping, flattening, etc. Figure 2.4 describes the testing results of techniques MNF-G, TFHE-DDA, Yah-scheme, and Than-scheme with time=60 means time >> 1 hour.

In [12], we noted a simple summary of the results linked to the execution speed. Table 2.4 displays these results which have been illustrated by Python utilizing a HP Laptop with the following characteristics: Processor Intel(R) Core(TM) i3-3110M CPU @ 2.40 GHz, 2 Core(s), 4 Logical Processor(s), 4 Go RAM. Utilizing a private key of 128 bits, a secret key of 1160 bits, and a text of 40 bits.

Scheme	Enc	Dec
ours	0.00014 s	0.0239 s
[123]	0.899 s	0.785 s
[122]	0.05 s	0.01 s
[124]	0.255 s	0.493 s

Table 2.4: DFE-P Processing time comparison

In [123], the authors conducted their experimentation on an Intel Xeon W3565 CPU for technique [125] of a private key of 128 bits, where in [122] a desktop computer of an Intel Core2 Duo E8500 CPU at 3.12 GHz with a secret key size $\eta = 1088$ bits, as in [124]. This technique was executed in C using the GNU

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

2.6.1 Spatial comparison

We did not focus here on ciphertext size characteristics in encryption techniques and compared them on this basis. This is because most encryption techniques work with the modulo operator which works in cyclic group, so the encryption is in public key order ($c = F \mod n$), and then the size of the ciphertext is in order of n. The size of n depends on the level of security, if the size of n increases, the level of security also increases.

Here, a simple comparison can be made between the asymmetrical techniques and the symmetrical technique. It is true that the latter is characterized by the small size of the encryption key, simply because it is not public, and therefore the size of the encrypted data will be less than in the asymmetric techniques; we speak for example about 256 bits vs. 1024 bit.

2.7 Conclusion

In the literature, the nature of cloud security has led researchers to cope with the homomorphic issue. Many proposed homomorphic schemes emit 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.

This chapter has covered security in the cloud at the compute level, the next chapter will deal with another aspect of security in the cloud which is the security at the data aggregation and storage levels.

Chapter 3

Blockchain for security concerns

To address another aspect of cloud security at the level of data aggregation and storage, we present two papers. A Consensus Algorithm for resist 51 Attacks, published in Applied Sciences journal. An improvement of the PoW protocol, published in AI-CSP conference.

Therefore, this chapter consists of the articles:"A Compute and Wait in PoW (CW-PoW) Consensus Algorithm for Preserving Energy Consumption, Applied Sciences Journal, 2021." and "A Novel Delegated Proof of Work Consensus Protocol, International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), November 2021.".

3.1 Introduction

Blockchain-as-a-service (BaaS) is the third-party creation and management of cloud-based networks for operating blockchain [126, 127, 128, 129, 130]. The use of blockchain technology has gone far beyond its most well-known use in cryptocurrency and has expanded to handle secure transactions of all kinds. BaaS is based on the SaaS and IaaS, it allows users to leverage cloud-based solutions to build, host, and run the blockchain applications and all related functions on the blockchain. IaaS offers encrypted data storage and distributed autonomous nodes that will make the consensus and/or mining (Figure 3.1). SaaS allows operating the blockchain applications. BaaS examples: Microsoft Azure, Amazon, R3, PayStand, etc.

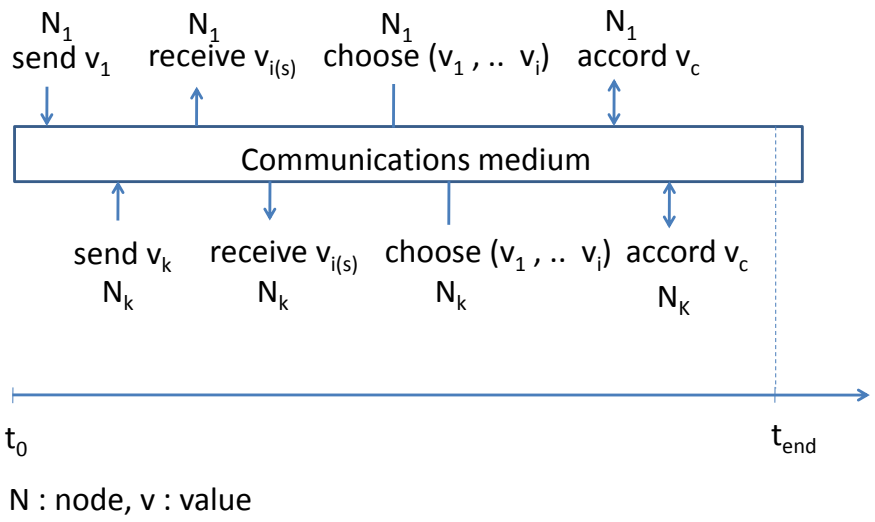

Figure 3.1: Consensus protocol in distributed system

3.1.1 Distributed computing

Distributed computing is a domain that explores distributed systems (DS) whose pieces are found on diverse networked computers dispersed over distinct geographies and communicate by dispatching data in order to attain a common purpose [131]. In this system with the lack of a global clock, the loss of a disconnected element in the system should not lead all others to fail. Each node of this environment has only a little and insufficient view of the system. Diverse software and hardware architectures are utilized for DS. In peer-to-peer (P2P), there are no particular devices that deliver services or control entire network resources. The obligations are evenly distributed among all devices which are called peers. Peers can act as both clients and servers.

In DS environment, the Byzantine Generals Problem (BGP) is a distributed calculation concern that was standardized by Robert Shostak, Leslie Lamport, and Marshall Pease [132] in 1982. The BGP is an analogy that in questioning the trustworthiness of communications and the integrity of the speakers. A set of parts operating together must certainly handle probable failures between them. The cause of failures is the presentation of incorrect or discordant information. The managing of faulty users is also named fault tolerance.In [133], Fischer, Lynch, and Paterson (FLP) have indicated that consensus can not be attained in a synchronous setting if even a third of the users are maliciously faulty. In an asynchronous system with a single faulty user, it is not feasible to guarantee that an accord can be attained.

FLP tells that accord is not always attained, but it does not say that it never

will be. This analysis involves asynchronous systems, when all processes work at unpredictable speeds [134]. A consensus can be achieved in practice, for a model by the non-perfect algorithms of Paxos Lamport [135] in obvious failures and no Byzantine faults. Lamport, Shostak, and Pease conducted in [132] that a consensus can be achieved if at least there are f faulty elements from $3f+1$. Under these situations, PoW has perfectly guaranteed the consensus, but its significant problem is the huge consumption of energy.

In this chapter, we present two consensus protocols for the cloud environment. We increase the security and minimize the energy consumption of autonomous cloud entities.

3.2 Consensus problem

A basic purpose of security in distributed environments, cloud computing, and multi-agent scenarios is to conduct general system reliability with the existence of faulty elements. This evidently demands the cooperation of honest parties in order to attain an agreement on a common objective. The users must consent to a single value, where each one must give its own value that is published to all the other parts. From all the suggested values, the nodes have to choose a single one. The accord stage is started by a leader element or at a predetermined duration. Many systems use consensus including blockchain, synchronization, smart grids, cloud computing, drone control, load balancing, etc.

3.2.1 Consensus conditions

The selection of a consensus strategy is the primary issue because it defines the security level and whole impact; so, we have to ensure the following matters:

- Accord: There is a common value selected by all honest users.
- Integrity: there is a single node decision without a changing selected value.
- Validity: The picked value by each node is one of the suggested values by others.
- Termination: The duration of a final decision must be in a finite time.

3.2.2 Number of nodes

Blockchain consensus is the method by which nodes in a decentralized network reach an agreement on the current state of the blockchain. Consensus is mostly concerned with achieving an honest majority in the face of a certain threshold of malicious actors and reaching finality; i.e., transactions are accurately processed and highly unlikely to ever be reversed. Blockchain consensus is generally designed around minimizing communication overhead in order to increase the upper bound on decentralization for stronger Byzantine fault tolerance and lower the time to finality for faster settlement.

based on the notion of competition and that only one can solve the mathematical problem (or prove something), there is no minimum number of nodes. For the maximum number, the propagation of the information in particular the data block in the network is an important factor to define this number. because the information must be propagated through the system before the next iteration begins.

3.3 Literature survey

Bitcoin [136] can present the most recognized blockchain implementation which is constructed in 2008 by parts working under the name "Satoshi Nakamoto". The basic infrastructure of this technology is a peer-to-peer network using the Internet [137]. Transactions define the exchanges between users, transactions are grouped jointly in blocks of size 1M which will be validated by miners. If the generated block is valid, it is time-stamped and then added to the blockchain. Once integrated into the blockchain, a block (transactions) can not be altered or deleted. A block contains transactions, a hash value (hash of transactions) utilized as an identifier, the hash of the last block, and the target which denotes a measurement of the work quantity that was demanded to build a block).

The major use of this mechanism is that of crypto-currencies like Bitcoin [138]. Despite its monetary facet, this decentralized system of storage can be used in numerous domains demanding secure interactions without passing by a centralizing element, systems demanding unfalsifiable traceability, usage founded on smart contracts, usage permitting the interaction of all sorts of services, etc. Each element in the system works autonomously respecting the defined rules. Identity managing protocols play the principal position in specifying users' organizing in the network.

Blockchain system comprises four classes of implementation. The network protocols, the protocols of distributed agreement, the autonomous framework that

is founded on smart contracts, and finally, the application is also called the interface [97]. Many consensus strategies are invented for each sort of blockchain. The most well-known protocol is Proof of Work (PoW), whose vision was first submitted in 1993 [139] by Moni Naor and Cynthia Dwork. In this protocol, the authors offered specifically a computational approach to combat spam and managing access to a shared resource.

The principal vision is to demand a user resolve a relatively difficult operation in order to use a resource. The demanded work must be challenging to accomplish for the requester, but easily checkable for a verifier. The phrase 'proof of work' has been appeared in 1999 by Ari Juels and Markus Jakobsson in [140]. For Bitcoin, the miner has to test a massive number of chances so that the hash of the given header is a suitable value.

Proof of Stake (PoS) algorithm appeared as a choice due to the elevated power consumption of PoW. PoS is used for the first time by Peercoin cryptocurrency in 2012 [141]. PoS demands the user to confirm ownership of a portion cyber money to be authorized to validate transactions. To bypass domination, numerous versions have been introduced for more complete consensus systems such as random share strategies considering the coin age like Peercoin cryptocurrency and relying on the acceleration [142] that is utilized by the ReddCoin cryptocurrency. The version which is viewed as equilibrated between decentralization, scalability, and security is Delegated Proof of Stake employed by the BitShares cryptomoney [143]. Miner picking is founded on polls; in which, the validator of transactions is randomly determined from 101 delegates who give the highest stakes.

Proof of Burn (PoB) protocol or Proof of Destruction [144], is a strategy equivalent to PoS where in PoS, each element is demanded to give some stake of money. If the user desires to exit the network he can recuperate his given stake. In PoB, the user implicates killing (burning) the given coins.

There are too multiple challenges that try to substitute "Work" in PoW protocol like Proof of eXercise (PoX) [145]. The challenge consists of solving a fundamental eXercise. The authors picked to solve a computation problem that is based on a matrix because matrix issues are linked with many practical problems. The miner has to find a solution for this equation:

$$X_1 \circ X_2 \circ ... \circ X_p = Y \tag{3.1}$$

with X_i and Y denote matrices, $\circ$ denotes an operator like product, sum, etc.

In Primecoin [146], another difficulty is proposed that consists of giving prime integers rather than giving a nonce.

Proof of Space [147] also called Proofs of Capacity (PoC) is performed between a verifier V and a prover P in two separate stages. The proposed idea is a verifier

give a pseudo-random file F of x GB and transmits it to a prover during an initialization stage. After, the verifier can request the prover to return a few bits of F at arbitrary places, confirming that P stores at least a big part of F. Unfortunately, the verifier always has to transmit to prover a massive file (100 GB), so this strategy can not be realized in practice.

Proof of Space-Time (PoST) [148] is an agreement strategy linked to PoC. PoST permits network parties to demonstrate that they have expended a space-time resource, the user has given storage space for a duration of time.

In Proof of Activity (PoA) protocol [149], miners commence with PoW where the built blocks do not include transactions, just simple templates including address and header information. After mining this blank block, the system swaps to PoS. To pick random validators to sign the transactions, the protocol utilizes header information. In PoA, there is again a big power consumption for mining

A random mining group picking to control 51% Attacks is suggested by [150]. The technique splits miners into many sets. Each user defines its group utilizing its wallet address and a hash function. After a block is constructed, its hash is utilized to define the next mining group. The possibility of a double-spend attack is decreased when increasing the number of groups where these groups are picked at random.

3.4 Proposed protocols

In this section, we will present two proposed protocols "A Compute and Wait in PoW Consensus Algorithm (CW-PoW)" and "A Novel Delegated Proof of Work Consensus Protocol (DPoW)".

3.4.1 CW-PoW

In [8], we proposed a consensus algorithm called CW-PoW. It aims to increase the security level in distributed systems and preserve energy consumption.

As illustrated by Figure 3.2, the general process in order to achieve an accord is split into different rounds. Each node establishes round number 1 and tries to give a solution for its transactions (block) like PoW protocol while a more inferior degree of challenge hardness is used. We assume that if the difficulty is X, the number of rounds is 1. In the CW-PoW, the challenge hardness is $X' = X/Y < X$ where the number of rounds is $Y > 1$. After giving a solution, the user broadcasts it and looks if the other nine (9) solutions of the current round are given in the

network or not. If yes, this user can commence the next round. If not, where the number of given solutions is less than 9, this candidate user stays waiting until receiving the rest solutions. The first user that give a solution in the last round will be the miner.

In the basic Proof of Work, the challenge consists of giving a nonce that realizes the following inequality: $Hash(Block + Nonce) < Target$. In CW-PoW protocol, the challenge of round i is defined by the illustrated inequality:

$$Hash(Block + IDRound_{i-1} + Nonce) < Target \tag{3.2}$$

The first time, the identifier (ID) of the round equals 1. then, *ID Round* equals the sum of solutions (nonces) given in the previous round. Thus, a new ID is calculated in each round where the users will work on it. With ten nonces to give per round, the following equation is performed:

$$IDRound_k = \sum_{i=1}^{10} (nonce_i \ of \ round_{k-1}) \tag{3.3}$$

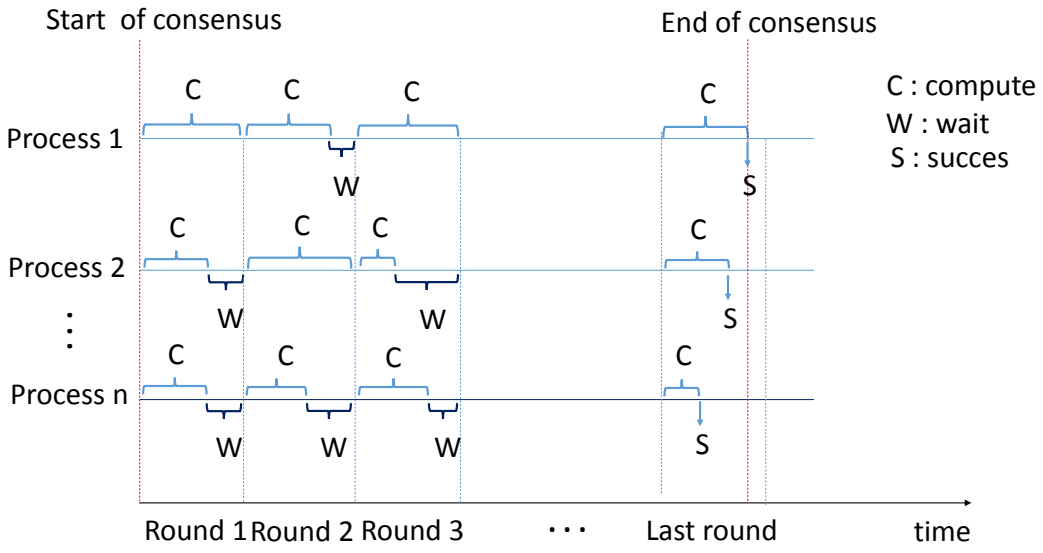

Figure 3.2: CW-PoW Consensus protocol.

Let $NbrR$ and $NbrS$ denote respectively the number of rounds and the number of solutions in each round. If two users give two nonces in the last round, other users have to compute the standard deviation of all given nonces (in all rounds) for each one of these two users. The miner will be the user that has the smallest standard deviation. If a user chooses to exit the challenger when the other users surpass it by x rounds, that decreases the energy consumption.

CW-PoW can be shown in Algorithm 3.

Algorithm 3 CW-PoW Consensus algorithm

Require: $NbrS,\ NbrR$
Ensure: *solution*

```
1: procedure Cons(NbrS, NbrR)
2:     sols ← 0
3:     roundID ← 0
4:     currentRound ← 1
5:     q ← Queue()
6:     blockID ← random()
7:     while currentRound <= NbrR do
8:         while solution not found do
9:             search for solution
10:        end while
11:        sols ← sols + 1
12:        if currentRound is NbrR then
13:            print(I am the winner) , Exit()
14:        end if
15:        Broadcast(currentRound, solution)
16:        put on(q, solution)
17:        while sols < NbrS do
18:            Nothing
19:        end while
20:        sols ← 0
21:        currentRound ← currentRound + 1
22:        RoundID ← ∑_{i=1}^{NbrS} q(i)
23:        Broadcast(currentRound, RoundID)
24:    end while
25: end procedure
```

3.4.2 Protocol proof

To achieve a consensus, four conditions must be realized which are accord, validity, integrity, and termination. In reality, we notice a fifth condition must be also admitted for each perfect agreement process. This requirement is chance equivalency or no domination. In the next paragraph, we illustrate how our proposal affirms these five requirements.

- Accord: We assume that one user finds the solution in the last round, all other users will agree on this winner. If there is more than one solution,

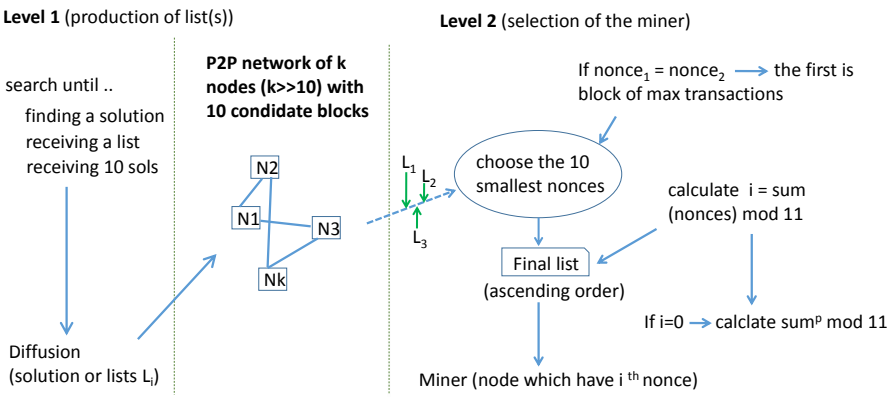

Figure 3.3: Proposed model for public blockchain

for each one, the standard deviation of all rounds must be calculated. The users will select the small standard deviation where this result is unique. Therefore, the selected winner is identical to all the honest users.

- Integrity: If a node N_x gets a valid solution S_1, the node publishes it with an exact timestamp. This solution is linked to a precise block (B_{i+1}) so $N_x(B_{i+1}, S_1)$. If N_x got another solution S_2, it will be published $(P_x(B_{i+1}, S_2))$; so, two solutions S_1 and S_2 of the identical block B_{i+1} have been broadcast. This issue can be solved by many methods like selecting the smallest one. Finally, one solution is selected for each user and the integrity requirement is achieved.

- Validity: All users must check the validity and originality of any broadcasted nonce where each user in the network is known by a unique public address.

- Termination: The approval on transactions is expressed by users by operating on making the next block, utilizing the hash value of the last block as a previous hash value [98]. PoW block generation time is evaluated at 10 minutes, this time is sufficient to propagate a block in the network. In our strategy, the number of rounds is increased and the difficulty of a challenge is decreased. To confirm the ending in a limited time, we have to manipulate the difficulty and number of rounds.

3.4.3 DPoW

In [13], we introduced two contributions, a modification of the basic PoW protocol for two objectives. The first objective is improving safety and presenting autonomy by lessening the domination issue of some users that have the most significant

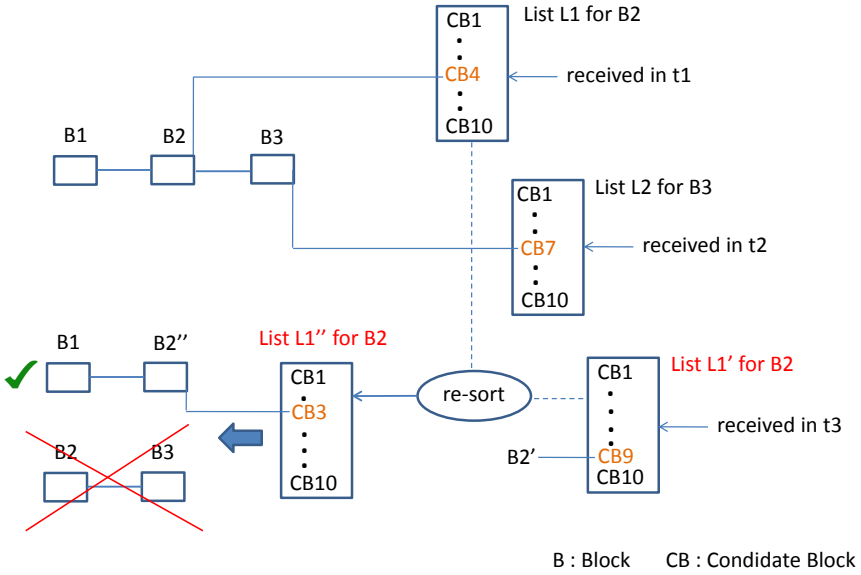

Figure 3.4: Forking issue in proposed algorithm

calculation power by picking ten candidate users in each iteration. The second objective is to reduce energy consumption by decreasing challenge difficulty. The other contribution is proposing an agreement architecture for private blockchain.

3.4.4 For public blockchain

Figure 3.3 illustrates that we have separated the consensus process into two steps; the first one consists of basic PoW using more inferior challenge difficulty. The other step consists of choosing one candidate user out of ten candidates, where a candidate user is a user that finds a solution. Each user will receive solutions from others, so, the computation persists until the user give its own nonce or the number of nonces received equal to ten. Thus, if a user obtained a list of ten solutions, it is not required to persist the computing.

If a user obtains more than one list that is distinct from each other, the user has to pick the ten smallest solutions, then construct a new list and broadcast it. If two solutions have the same values, the first position is given to the block that has more transactions.

After constructing a list, in the second step, each user has to pick randomly a candidate using Equation 3.4.

$$v = (\sum_{i=1}^{10} S_i) \mod 11 \tag{3.4}$$

with S_i denotes solution i, which is linked to the blocks$_i$. That involved $v \in \{0, 10\}$, if $v > 0$, the miner is the proprietor of the vth Block. In another case where $v = 0$, the user must computes: $v' = v^p \mod 11$. Finally, one winner will be selected, and all users will give the same candidate block.

Blockchain evolution:
This is a Forking issue; in the original PoW, processes ever take the most extended chain as a correct blockchain. If there are at the same time two distinct versions, each user will operate on the first one he received with saving the other one. If the saved version becomes more extended, the user change to the longer version [98]. Nevertheless, in our proposal, the user returns to the block that is earlier added (block $B2$ in Figure 3.4); then, the user withdraws all the blocks on which operated and arrives back to the same point (block $B2$). Finally, the full system will keep the same blockchain.

The distinction between one user and another may be one or two blocks which are the blocks that were constructed at the propagation time, those blocks will be withdrawn and the users will commence operating a new list which ensures system auto-synchronization. Also, If the users operate on block N, the new solutions of block $N - 2$ are ignored in the aim to bypass disrupting the system.

3.4.5 For private blockchain

Algorithm 4 Consensus algorithm

Require: P: prime number, N: number of nodes
Ensure: miner

function CONS
  generate m
  generate k
  $c_1 \leftarrow m \times k \mod p$
  $c_2 \leftarrow k^k \mod p$
  broadcast c_1, c_2
  while number of received c_i < N **do**
    receive c_i
  end while
  broadcast k
  while number of received k_i < N **do**
    receive k_i
  end while
  $m_i \leftarrow$ decrypt all c_{1i} using k_i
  $v \leftarrow \sum_{i=1}^{N} m_i \mod p$
  select miner using v
  return miner
end function

In private blockchain where there is a small number of users, the number of candidate users equals the number of system users, so is not required to destroy energy and time in proof of work (the first step). We have used here two steps as shown in Figure 3.5. In the first one that is list composition, each user P_i gives a random number m_i which denotes the selection value, this value is exact for this iteration, also a random private key k_i to encrypt m_i using Equation 3.5.

$$c_i = m_i \times k_i \mod p \tag{3.5}$$

with p is a public prime number. The value m to prevent faulty nodes to manipulate the selection result. To prevent nodes to modify its private key k or modify the picking value where $\exists\ m'\ and\ k'\ :\ m \times k \mod p = m' \times k' \mod p$, the user must share c_i and c'_i with:

$$c'_i = k_i^{k_i} \mod p \tag{3.6}$$

Each user get all c_i and c'_i of all other users, let List = (c_i, c'_i), $i \in \{1, n\}$, after completing the list, the picking step starts.

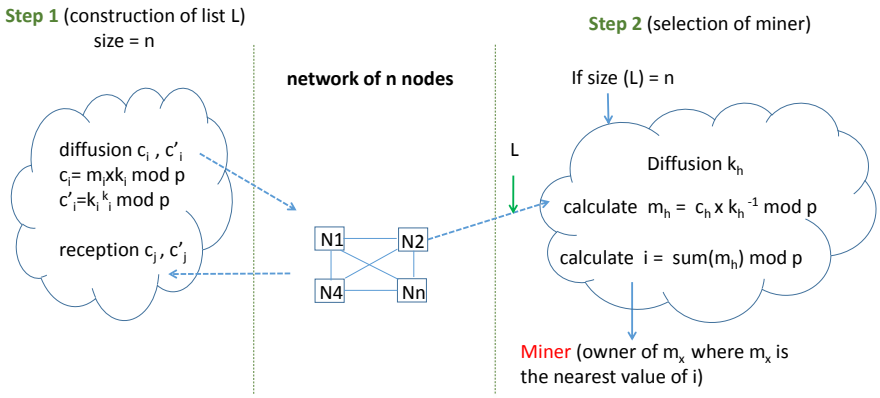

Figure 3.5: Proposed model for private blockchain

In the selection step, each user reveals its private key k. Thus, any user in the system can compute all the m_i by decrypting the c_i where $m_i = c_i \times k_i^{-1} \mod p$. Therefore, the users can compute v using Equation 3.7.

$$v = (\sum_{i=1}^{n} m_i) \mod p \tag{3.7}$$

The miner user is the one that possesses the selection value m which is nearest to the calculated value v. If one of the users has failed after list construction where all the users wait for k_i, each user has to wait for a precise time, then return to the first phase again.

3.5 Cryptanalytic attacks

For CW-PoW
The safety level of PoW is founded on the challenge difficulty. Nevertheless, it stays vulnerable against 51% attacks [151]. Concerning the domination issue, there is always only one candidate node which has the most significant calculation power. We decreased the calculation difficulty to allow 10 nodes to be candidates, so the probability of the greatest power node to be a winner is 10% rather than 100% because no user can maintain the standard deviation of its given nonces. Thus, the dominance is decreased.

When dominance is diminished, the 51% attack possibility will also be decreased. The pool can not operate voluntarily to get only one nonce, it must pass by different rounds. In each one, the user must transmit the current round identifier which is computed using the found nonces of the previous round as shown in

Equation (3.3). That stage will delay the process of constructing another branch with the aim to perform a double-spending attack.

About Sybil attack, it is unworkable if the attacker has less than 50% of the system's calculation power. Also, the agreement strategy does not control an adversary to disrupt the system by constructing many false identities. The presented model mixes two methods, multi-rounds, and standard deviation. Compared with the original PoW, it is hard for a malicious node to be a miner. If a malicious node tries to make another longer chain, the first strategy (multi-rounds) will delay its job, and the second one (standard deviation) will reduce its opportunities.

For DPoW
The accord in a distributed system is a difficult process, consensus strategies have to treat many issues including the following:

- **Energy:** After overcoming many barriers like the capacity of blocks represented in the number of transactions. There are other issues like the huge energy consumption. According to [152] in 2020 which examines the energy consumption by blockchain, the authors have discovered that the consumption of electricity was between 60 and 125 TWh per year for Bitcoin; knowing that the annual consumption of electricity by countries such as Austria is 75 TWh and Norway is 125 TWh.

  The Target denotes a value utilized in mining, this value is updated every 2016 blocks. In our proposal, we maximize the target value with the aim to reduce the calculation time, and thus we will get an energy consumption improvement. To recuperate this lowered compute time and to guarantee that introducing a new block is accomplished in 10 minutes, we present a second element which is the propagation of generated solutions in the network. For example, the difficulty is decreased to 50%, the nodes give a nonce in 5 minutes, and during the rest time, it waits for the other 9 nonces. Therefore, energy consumption is decreased.

- **Security:** In the last paragraph, we noticed that to improve the safety level, the PoW uses a high degree of challenge, but it is also vulnerable to 51% attack [151]. Concerning dominance, we reduce the difficulty and get ten candidate users, so the probability that a user wins is 10% rather than 100% due to the selection approach which is pseudo-random as shown in Equation 3.4). Therefore, the dominance has been decreased by 90%.

  About 51% attack, it is hard to be performed because the integration of blocks into blockchain follows Equation 3.4 and does not use the longest chain strategy. This equation uses part of luck, when the malicious pool makes ten candidate blocks, there are ten other candidate blocks built in the system. In the proposed strategy, we assume that the new $block_{pool}$ is

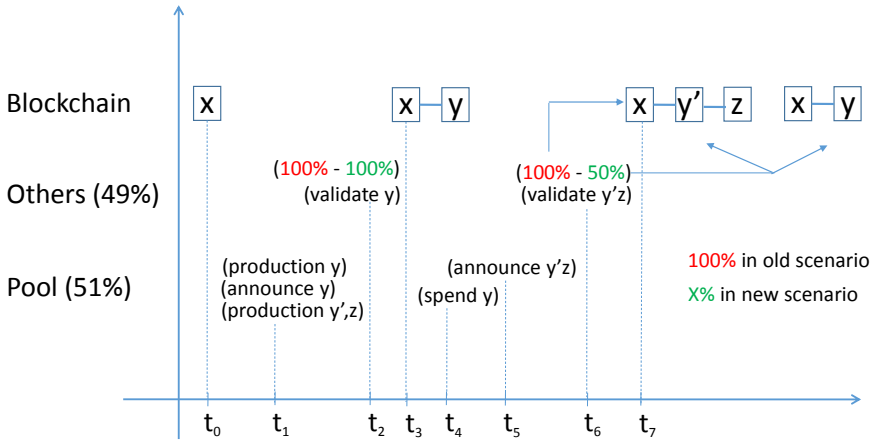

Figure 3.6: Comparison between old and new scenario

the winner, the system will not accept the new branch of the pool because the current block has 20 candidates. Thus, the rate for $block_{pool}$ to win is 50%; so, the double-spending attack has reduced by 50% (Figure 3.6).

Speaking of the Sybil attack, the semi-random choice of the winners from several candidate nodes will reduce the possibility of this attack, especially if these candidate nodes are required to prove work first.

3.6 Experimental results

In CW-PoW experimentation, each node is simulated by a process, and multi-process programming is implemented. In many tests, we created a different number of processes, each of them beginning by creating a random number $blockID \leftarrow random()$, that is used as an ID of the candidate block which the process is working on. After, each process fellows the instructions of our proposed algorithm (Algorithm 3). Selecting five rounds and ten processes, the test took about 5 minutes. Figure 3.7 shows the obtained result. The experimentation has been done in Python using an HP Laptop that have the following designations: Processor Intel(R) Core(TM) i3-3110M CPU @ 2.40 GHz, 4 Go RAM.

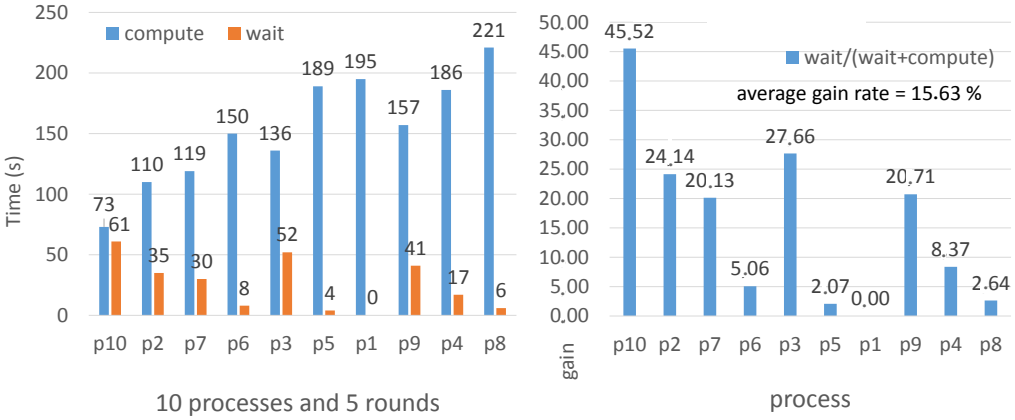

Figure 3.7: Compute and wait example execution

The experiment has been performed tens of times in each test. In Figure 3.8, the number of both rounds and processes is equal to 5, so the gain was 12.07 by taking the most frequent result.

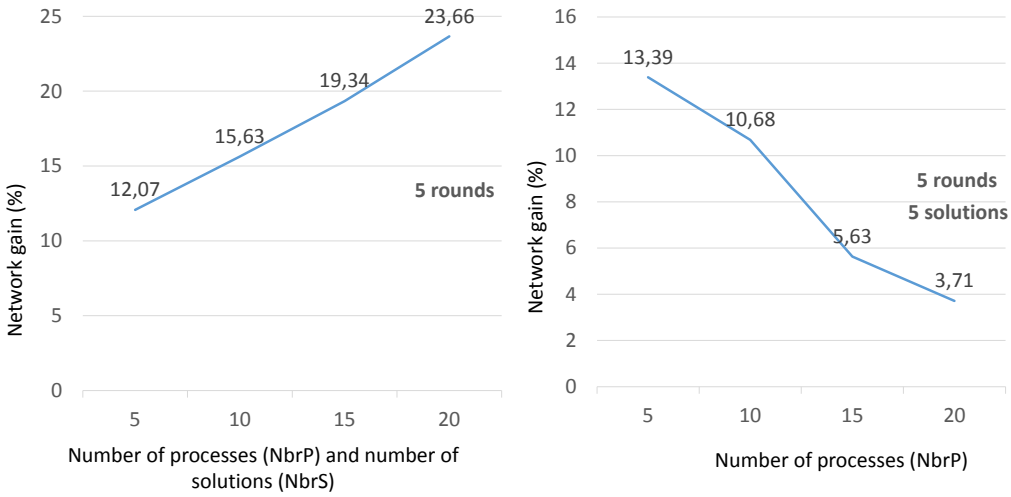

Figure 3.8: The impact of NbrP for fixed NbrR.

The gain formal definition: The wait time is considered as a gain, so this gain rate is formulated as follows: *gain rate = total wait time/total execution time.*

$$gain\ rate = wait\ time/(wait\ time + compute\ time) \tag{3.8}$$

We put $C_{i,x}$ and $W_{i,x}$) be the compute and wait time of the node i in round x. The gain of the node i is:

$$gain_i = (\sum_{j=1}^{NbrR} W_{i,j})/(\sum_{j=1}^{NbrR} W_{i,j} + \sum_{j=1}^{NbrR} C_{i,j}) \tag{3.9}$$

Therefore, the network gain rate is:

$$Gain = (\sum_{i=1}^{NbrP} gain_i)/NbrP \tag{3.10}$$

Figure 3.7 shows a test where the number of rounds is $NbrR = 5$, the number of processes is $NbrP = 10$, and the number of solutions is $NbrS = 10$. Node number 10 is the fast one due to its minimum execution time and its maximum waiting time, that is why it has the highestwith a gain rate equals to 45.52%.

For process p_1, the gain is 0 / (0 + 195) = 0 because it is the last one to find a nonce; finally, the average gain rate of the whole network was 15.63%.

Compute and wait implementation: There are two principal characteristics that must be tolerated, the number of rounds $NbrR$ and the number of solutions $NbrS$.

Lemma 2. *If* $NbrR > 1$ *then* $Gain > 0$

Proof. The process uses a random hash function (Block + Nonce) < Target; also, the speed of each process is incidental, so:

$$C_{i,x} \neq C_{j,x},\ \forall i, j \in \{1, .., NbrP\},\ \forall x \in \{1, .., NbrR\} \tag{3.11}$$

where $0 < x <= NbrR$. According to Equation (3.11), $|C_{i,x} - C_{j,x}| > 0$, so either $W_{i,x} > 0$ or $W_{j,x} > 0$, which implies that $W_{i,x}/(W_{i,x} + C_{i,x}) > 0$ or $W_{i,x}/(W_{j,x} + C_{j,x}) > 0$. So, $Gain > 0$. □

The rate gain in the network is $(G_{nw}) = (g_1 + g_2 + \ldots g_{NbrR})/NbrR$ where g_i denotes the gain of round i. Knowing that the gain of the last round equal to 0

because there is no wait. The execution times of each process is random in each round, therefore, $g_1 \approx g_2 \approx \ldots g_{NbrR-1}$ and then $G_{nw} = (NbrR - 1) \times g_1/NbrR$

So:

$$G_{nw} = g_1 - (g_1/NbrR) \tag{3.12}$$

Figure 3.9 demonstrate how the gain converges to a constant G if the number of rounds increases $NbrR = 5$, 10, 15, 20, 25.

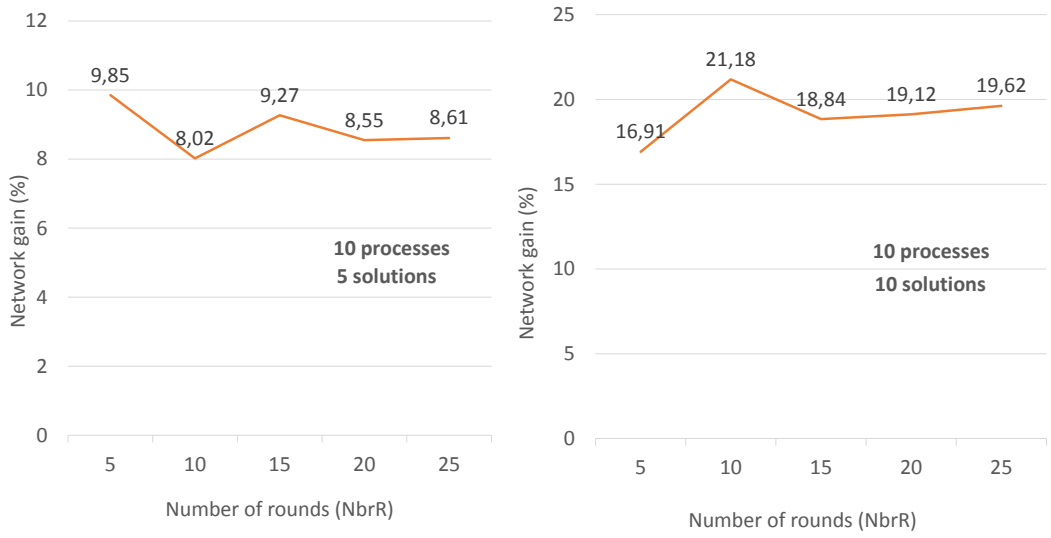

Figure 3.9: The influence of NbrR, case B

We cannot use a different umber of solutions without using a multi-round system. Figure 3.10 demonstrate the increase in gain rate if $NbrS$ increases

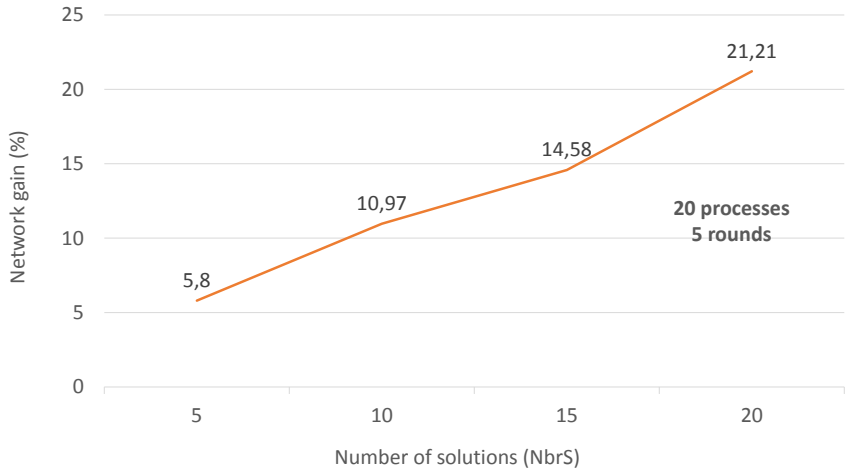

Figure 3.10: The influence of NbrS on the network gain rate

If the process exits the competition when $NbrS$ of the next round is provided, the gain will be a maximum (Table 3.1).

NbrR	**NbrS**	**NbrP**	**Gain Rate (%)**
5	5	10	55
5	5	20	60
5	5	30	70
5	5	100	90

Table 3.1: The gain rate with leaving competition

3.7 Conclusion

In order to raise the level of security in the cloud and distributed systems in terms of collecting and storing data, as well as giving autonomy, flexibility, and confidence to the cloud entities, we have developed consensus algorithms: CW-PoW and DPoW. In this chapter, we have explained each of them in detail.

For DPoW, We have theoretically shown its efficiency against the Sybil attack and 51% attack with a rate higher than the original PoW up to 50% by selecting ten candidate nodes at each iteration. Also, we have introduced a model accord suitable to a private blockchain while the number of users is known in advance.

The presented model is founded on a lightweight encryption technique. We have theoretically displayed that the results of this accord cannot be manipulated or falsified. Also, it is impossible to predict which user will be a miner.

We have examined the validity of the proposed CW-PoW according to the four requirements of an accord. We have demonstrated the energy gain performed by this algorithm to reach a reduction of 15.63% using five rounds and to reach a reduction of 19.91% using ten rounds. We have conducted many scenarios selecting each test, with a modification of three characteristics (rounds number, process number, and the number of solutions per round). We have analyzed individually the impact of each characteristic on the consumed energy, and also, the impact of introducing these elements in comparison with the original PoW. We have analyzed the algorithm's hardness against the most well-known attacks such as 51% and the Sybil attack.

Conclusion

Nowadays, Cloud computing offers several services to users to facilitate their work and improve their lives. Users collect and share confidential data that must be kept and managed securely. Cloud computing delivers an agreeable medium that authorizes clients to access and re-access data. Nevertheless, the cloud is an untrusted part that can disclose data when decoded for processing by entities.

This work is interested in this Cloud challenge. So, the manuscript fits into the context of the problem of data security. On the one hand, we have discussed the issue of processing encrypted data. Often data is encrypted before being transmitted to the cloud, this last must have a clear version of data to be able to perform computations, that increase security and privacy menaces if the cloud is considered untrusted. Demanding clients to conduct the calculations after decrypting received data from the cloud, then encrypting the got results before sending them back is not a practical idea in distributed multi-tenant systems. Homomorphic encryption mechanisms make it possible to provide a solution for handling encrypted data and to make cloud entities autonomous.

On the other hand, we have proposed solutions for securing autonomous cloud entities that collect data (or generally process information) as part of Blockchain-as-a-service (BaaS). These solutions are in the form of consensus algorithms.

In the introduction, we put the reader in the context of the thesis by talking generally about distributed systems and cloud computing. Then we started getting into the subject by talking about security and the autonomous environment. After mentioning the main contributions, we sealed the introduction by organizing the manuscript and making references to the publications.

We started in the first chapter with mathematical concepts and definitions; then, we defined the problems on which many encryption techniques are based, these are factorization and discrete logarithm. Then, we are focused on the concept of cryptography, and some notions under this concept such as the type of encryptions (symmetric, asymmetric), quantum cryptography, and electronic signature schemes. Subsequently, the types of cryptosystems and the data exchange protocols were defined. The next point was about cloud computing, its features, types, security, and threats. The last point of this chapter was on the blockchain, we mentioned its uses, characteristics, and potential attacks on this technology.

In the second chapter, we detailed our contributions to encryption. These contributions address the problem of the confidentiality of data stored in the Cloud by proposing two homomorphic encryption techniques (MNF-G and DFE-PR). The objective was to make cloud computing entities autonomous by letting them perform different operations (addition and multiplication) on encrypted data. These techniques are simple to apply because they are based on linear and modular expressions. We have mathematically validated their homomorphic properties and made a security analysis of these schemes. The implementations demonstrate their feasibility and their great efficiency in terms of execution time compared with other schemes.

In the last chapter, we introduced two other contributions whose goal was to increase the security of autonomous Cloud entities. Indeed, two consensus protocols CW-PoW and DPoW have been proposed. Their validity and feasibility have been demonstrated; also, we made cryptanalytic attacks on them. The results of experiments have shown their effectiveness. To conclude, this thesis has allowed us to examine a wide range of concepts, models, and technologies in the fields of data security and the Cloud. We have provided new techniques and protocols to achieve our goal of securely managing Cloud entities in a distributed system. We have also shown that the proposed work joins a rich and encouraging research theme.

Outlook

- We would like to study the robustness of the proposed techniques/protocols against other known attacks.
- We want to enlarge the notion of autonomy in other domains like manufacturing.
- We intend to make a thorough study to implement MNF-G and DFE-PR in an IoT environment.
- We intend to combine homomorphic encryption with machine learning so that we can perform analysis and prediction on the encrypted data.

Bibliography

Bibliography

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

[2] Yagoub Mohamed Amine. *Une Approche Basée Agent Pour La Sécurité Dans Le Cloud Computing*. PhD thesis, Université Mohamed Khider - Biskra, 2019.

[3] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[4] Zaid Kartit. Contribution à la sécurité du cloud computing: Application des algorithmes de chiffrement pour sécuriser les données dans le cloud storage. *thesesenafrique.imist.ma*, 2016.

[5] Mladen A Vouk. Cloud computing–issues, research and implementations. *Journal of computing and information technology*, 16(4):235–246, 2008.

[6] Microsoft cloud computing. `https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/`. Accessed: 2022-03-13.

[7] 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.

[8] Mostefa Kara, Abdelkader Laouid, Muath AlShaikh, Mohammad Hammoudeh, Ahcene Bounceur, Reinhardt Euler, Abdelfattah Amamra, and Brahim Laouid. A compute and wait in pow (cw-pow) consensus algorithm for preserving energy consumption. *Applied Sciences*, 11(15):6750, 2021.

[9] Mostefa Kara, Abdelkader Laouid, Muath AlShaikh, Ahcène Bounceur, and Mohammad Hammoudeh. Secure key exchange against man-in-the-middle attack: Modified diffie-hellman protocol. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, 7(3):380–387, 2021.

[10] MOSTEFA KARA, ABDELKADER LAOUID, AHCÈNE BOUNCEUR, MOHAMMAD HAMMOUDEH, and MUATH ALSHAIKH. Perfect confidentiality through unconditionally secure homomorphic encryption using otp with a single pre-shared key. *Journal of Information Science and Engineering*, 39(1):183–195, 2023.

[11] Mostefa Kara, Abdelkader Laouid, Mohammad Hammoudeh, Muath Alshaikh, and Ahcene Bounceur. Proof of chance: A lightweight consensus algorithm for the internet of things. *IEEE Transactions on Industrial Informatics*, pages 1–1, 2022.

[12] 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.

[13] Mostefa Kara, Abdelkader Laouid, Ahcene Bounceur, Farid Lalem, Muath AlShaikh, Romaissa Kebache, and Zaoui Sayah. A novel delegated proof of work consensus protocol. In *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*, pages 1–7. IEEE, 2021.

[14] Mostefa Kara, Abdelkader Laouid, Ahcene Bounceur, Mohammad Hammoudeh, Muath Alshaikh, and Romaissa Kebache. Semi-decentralized model for drone collaboration on secure measurement of positions. In *The 5th International Conference on Future Networks & Distributed Systems*, pages 64–69, 2021.

[15] Aissaoua Habib, Abdelkader Laouid, and Mostefa Kara. Secure consensus clock synchronization in wireless sensor networks. In *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*, pages 1–6. IEEE, 2021.

[16] Khaled Chait, Abdelkader Laouid, Lamri Laouamer, and Mostefa Kara. A multi-key based lightweight additive homomorphic encryption scheme. In *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*, pages 1–6. IEEE, 2021.

[17] Mohammed Elhabib Kahla, Mounir Beggas, Abdelkader Laouid, Mostefa Kara, and Muath AlShaikh. Asymmetric image encryption based on twin message fusion. In *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*, pages 1–5. IEEE, 2021.

[18] Pierre-Alain Fouque. *Le partage de clés cryptographiques: Théorie et Pratique*. PhD thesis, Paris 7, 2001.

[19] Cyrielle Feron. *PAnTHErS : un outil d'aide pour l'analyse et l'exploration d'algorithmes de chiffrement homomorphe*. Theses, ENSTA Bretagne - École nationale supérieure de techniques avancées Bretagne, November 2018.

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

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

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

[23] 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.

[24] 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.

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

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] Stéphane Jacob. *Protection cryptographique des bases de données: conception et cryptanalyse*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2012.

[32] 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.

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

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

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

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

[37] Gilles Brassard et al. *Modern cryptology: A tutorial*, volume 325. Springer, 1988.

[38] 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.

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

[40] 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.

[41] 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.

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

[43] 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.

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

[45] 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.

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

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[56] Boaz Barak and Mohammad Mahmoody-Ghidary. Merkle puzzles are optimal—an o (n 2)-query attack on any key exchange from a random oracle. In *Annual International Cryptology Conference*, pages 374–390. Springer, 2009.

[57] Antoine Joux. The weil and tate pairings as building blocks for public key cryptosystems. In *International Algorithmic Number Theory Symposium*, pages 20–32. Springer, 2002.

[58] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[59] Boris Korzh, Charles Ci Wen Lim, Raphael Houlmann, Nicolas Gisin, Ming Jun Li, Daniel Nolan, Bruno Sanguinetti, Rob Thew, and Hugo Zbinden. Provably secure and practical quantum key distribution over 307 km of optical fibre. *Nature Photonics*, 9(3):163–168, 2015.

[60] Charles H Bennett. Quantum cryptography using any two nonorthogonal states. *Physical review letters*, 68(21):3121, 1992.

[61] Valerio Scarani, Antonio Acin, Grégoire Ribordy, and Nicolas Gisin. Quantum cryptography protocols robust against photon number splitting attacks for weak laser pulse implementations. *Physical review letters*, 92(5):057901, 2004.

[62] Priti Kumari and Parmeet Kaur. A survey of fault tolerance in cloud computing. *Journal of King Saud University-Computer and Information Sciences*, 33(10):1159–1176, 2021.

[63] Shyam Patidar, Dheeraj Rane, and Pritesh Jain. A survey paper on cloud computing. In *2012 second international conference on advanced computing & communication technologies*, pages 394–398. IEEE, 2012.

[64] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[65] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. *faculty.winthrop.edu*, 2011.

[66] Laura Savu. Cloud computing: Deployment models, delivery models, risks and research challenges. In *2011 International Conference on Computer and Management (CAMAN)*, pages 1–4. IEEE, 2011.

[67] Mohammad Ubaidullah Bokhari, Qahtan Makki Shallal, and Yahya Kord Tamandani. Cloud computing service models: A comparative study. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 890–895. IEEE, 2016.

[68] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen, and Zhenghu Gong. The characteristics of cloud computing. In *2010 39th International Conference on Parallel Processing Workshops*, pages 275–279. IEEE, 2010.

[69] Peter Buxmann, Thomas Hess, and Sonja Lehmann. Software as a service. *Wirtschaftsinformatik*, 50(6):500–503, 2008.

[70] Eman AbdElfattah, Mohamed Elkawkagy, and Ashraf El-Sisi. A reactive fault tolerance approach for cloud computing. In *2017 13th international computer engineering conference (ICENCO)*, pages 190–194. IEEE, 2017.

[71] Lakshmi Prasad Saikia and Yumnam Langlen Devi. Fault tolerance techniques and algorithms in cloud computing. *International Journal of Computer Science & Communication Networks*, 4(1):01–08, 2014.

[72] Yucong Duan, Guohua Fu, Nianjun Zhou, Xiaobing Sun, Nanjangud C Narendra, and Bo Hu. Everything as a service (xaas) on the cloud: origins, current and future trends. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 621–628. IEEE, 2015.

[73] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy and survey of cloud computing systems. In *2009 Fifth International Joint Conference on INC, IMS and IDC*, pages 44–51. Ieee, 2009.

[74] Saurabh Singh, Young-Sik Jeong, and Jong Hyuk Park. A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications*, 75:200–222, 2016.

[75] Mehdi Nazari Cheraghlou, Ahmad Khadem-Zadeh, and Majid Haghparast. A survey of fault tolerance architecture in cloud computing. *Journal of Network and Computer Applications*, 61:81–92, 2016.

[76] Dimitrios Zissis and Dimitrios Lekkas. Addressing cloud computing security issues. *Future Generation computer systems*, 28(3):583–592, 2012.

[77] M Talaat, Abdulaziz S Alsayyari, Adel Alblawi, and AY Hatata. Hybrid-cloud-based data processing for power system monitoring in smart grids. *Sustainable Cities and Society*, 55:102049, 2020.

[78] Subashini Subashini and Veeraruna Kavitha. A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11, 2011.

[79] Adam A Noureddine and Meledath Damodaran. Security in web 2.0 application development. In *Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services*, pages 681–685, 2008.

[80] Boualem Benatallah, Fabio Casati, Florian Daniel, and Jin Yu. Mashups, saas, and cloud computing: Evolutions and revolutions in the integration landscape. *Document based on Tutorial at ICDE*, 2009.

[81] Pradnyesh Rane. Securing saas applications: a cloud security perspective for application providers. *Information Security Management Handbook*, 5:301–311, 2010.

[82] Rohit Bhadauria, Rituparna Chaki, Nabendu Chaki, and Sugata Sanyal. Security issues in cloud computing. *Acta technica corviniensis-bulletin of engineering*, 7(4), 2014.

[83] Deyan Chen and Hong Zhao. Data security and privacy protection issues in cloud computing. In *2012 International Conference on Computer Science and Electronics Engineering*, volume 1, pages 647–651. IEEE, 2012.

[84] Florin OGIGĂU-NEAMŢIU et al. Cloud computing security issues. *Journal of Defense Resources Management (JoDRM)*, 3(2):141–148, 2012.

[85] Tareq Ahram, Arman Sargolzaei, Saman Sargolzaei, Jeff Daniels, and Ben Amaba. Blockchain technology innovations. In *2017 IEEE Technology Engineering Management Conference (TEMSCON)*, pages 137–141, 2017.

[86] Sherif F Fahmy. Blockchain and its uses. In *Arab Academy for Science and Technology and Maritime Transport*. Sheraton, 2018.

[87] Pietro Danzi, Marko Angjelichinoski, Čedomir Stefanović, and Petar Popovski. Distributed proportional-fairness control in microgrids via blockchain smart contracts. In *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 45–51. IEEE, 2017.

[88] Paul Gordon, Marni Hood, and Henry Materne-Smith. Crypto-contracts: The coming of the blockchain revolution. *Bulletin (Law Society of South Australia)*, 39(3):34–35, 2017.

[89] Melanie Swan. *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.

[90] Lewis Rinaudo Cohen, Lee Samuelson, and Hali Katz. How securitization can benefit from blockchain technology. *The Journal of Structured Finance*, 23(2):51–54, 2017.

[91] Dinesh Verma, Nirmit Desai, Alun Preece, and Ian Taylor. A block chain based architecture for asset management in coalition operations. In *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR VIII*, volume 10190, page 101900Y. International Society for Optics and Photonics, 2017.

[92] Daniel Augot, Hervé Chabanne, Olivier Clémot, and William George. Transforming face-to-face identity proofing into anonymous digital identity using the bitcoin blockchain. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 25–2509. IEEE, 2017.

[93] Akshay Bakre, Nikita Patil, and Sakshum Gupta. Implementing decentralized digital identity using blockchain. *International Journal of Engineering Technology Science and Research*, 4(10):379–385, 2017.

[94] Sead Muftic. Blockchain identity management system based on public identities ledger, April 25 2017. US Patent 9,635,000.

[95] Devrim Unal, Mohammad Hammoudeh, and Mehmet Sabir Kiraz. Policy specification and verification for blockchain and smart contracts in 5g networks. *ICT Express*, 6(1):43–47, 2020.

[96] Yang Xiao, Ning Zhang, Wenjing Lou, and Y Thomas Hou. A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465, 2020.

[97] Wenbo Wang, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. A survey on consensus mechanisms and mining strategy management in blockchain networks. *IEEE Access*, 7:22328–22370, 2019.

[98] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

[99] Shijie Zhang and Jong-Hyouk Lee. Double-spending with a sybil attack in the bitcoin decentralized network. *IEEE Transactions on Industrial Informatics*, 15(10):5715–5722, 2019.

[100] Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of ddos attacks against bitcoin mining pools. In *International Conference on Financial Cryptography and Data Security*, pages 72–86. Springer, 2014.

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

[102] 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.

[103] 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.

[104] 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.

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

[106] 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.

[107] 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.

[108] 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.

[109] 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.

[110] 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.

[111] 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.

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

[113] 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.

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

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

[116] 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.

[117] 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.

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

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

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

[121] 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.

[122] 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.

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

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

[125] 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.

[126] Mayra Samaniego, Uurtsaikh Jamsrandorj, and Ralph Deters. Blockchain as a service for iot. In *2016 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, pages 433–436. IEEE, 2016.

[127] Weilin Zheng, Zibin Zheng, Xiangping Chen, Kemian Dai, Peishan Li, and Renfei Chen. Nutbaas: A blockchain-as-a-service platform. *Ieee Access*, 7:134422–134433, 2019.

[128] Daming Li, Lianbing Deng, Zhiming Cai, and Alireza Souri. Blockchain as a service models in the internet of things management: systematic review. *Transactions on Emerging Telecommunications Technologies*, page e4139, 2020.

[129] Qinghua Lu, Xiwei Xu, Yue Liu, Ingo Weber, Liming Zhu, and Weishan Zhang. ubaas: A unified blockchain as a service platform. *Future Generation Computer Systems*, 101:564–575, 2019.

[130] SONG Jie, Pengyi ZHANG, Mohammed ALKUBATI, BAO Yubin, and YU Ge. Research advances on blockchain-as-a-service: Architectures, applications and challenges. *Digital Communications and Networks*, 2021.

[131] Hector Garcia-Molina. Elections in a distributed computing system. *IEEE Transactions on Computers*, C-31:48–59, 1982.

[132] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the Works of Leslie Lamport*, pages 203–226. ACM, 2019.

[133] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[134] John Turek and Dennis Shasha. The many faces of consensus in distributed systems. *Computer*, 25(6):8–17, 1992.

[135] Leslie Lamport. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*, pages 277–317. ACM, 2019.

[136] Satoshi Nakamoto and A Bitcoin. A peer-to-peer electronic cash system. *Bitcoin.–URL: https://bitcoin. org/bitcoin. pdf*, 4, 2008.

[137] Vipin Kumar Rathi, Vinay Chaudhary, Nikhil Kumar Rajput, Bhavya Ahuja, Amit Kumar Jaiswal, Deepak Gupta, Mohamed Elhoseny, and Mohammad Hammoudeh. A blockchain-enabled multi domain edge computing orchestrator. *IEEE Internet of Things Magazine*, 3(2):30–36, 2020.

[138] Mohammed Mudassir, Shada Bennbaia, Devrim Unal, and Mohammad Hammoudeh. Time-series forecasting of bitcoin prices using high-dimensional features: a machine learning approach. *Neural Computing and Applications*, pages 1–15, 2020.

[139] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992.

[140] Markus Jakobsson and Ari Juels. Proofs of work and bread pudding protocols. In *Secure information networks*, pages 258–272. Springer, 1999.

[141] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19:1, 2012.

[142] Larry Ren. Proof of stake velocity: Building the social currency of the digital age. *Self-published white paper*, 2014.

[143] Fabian Schuh and Daniel Larimer. Bitshares 2.0: general overview. *accessed June-2017.[Online]. Available: http://docs. bitshares. org/downloads/bitshares-general. pdf*, 2017.

[144] Kostis Karantias, Aggelos Kiayias, and Dionysis Zindros. Proof-of-burn. In *International Conference on Financial Cryptography and Data Security*, pages 523–540. Springer, 2020.

[145] Ali Shoker. Sustainable blockchain through proof of exercise. In *IEEE 16th International Symposium on Network Computing and Applications (NCA)*, pages 1–9. IEEE, 2017.

[146] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. *July 7th*, 1(6), 2013.

[147] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Annual International Cryptology Conference*, pages 585–605. Springer, 2015.

[148] Tal Moran and Ilan Orlov. Simple proofs of space-time and rational proofs of storage. In *Annual International Cryptology Conference*, pages 381–409. Springer, 2019.

[149] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y. *ACM SIGMETRICS Performance Evaluation Review*, 42(3):34–37, 2014.

[150] Jaewon Bae and Hyuk Lim. Random mining group selection to prevent 51% attacks on bitcoin. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 81–82. IEEE, 2018.

[151] Ning Shi. A new proof-of-work mechanism for bitcoin. *Financial Innovation*, 2(1):1–8, 2016.

[152] Johannes Sedlmeir, Hans Ulrich Buhl, Gilbert Fridgen, and Robert Keller. The energy consumption of blockchain technology: beyond myth. *Business & Information Systems Engineering*, 62(6):599–608, 2020.