الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية التكنولوجيا

قسم: الاعلام الالي

Master's thesis

Specialty: Sécurité Informatique et Cryptographie

Theme

Comparative Study of Cryptography Algorithms

Presented by:

Alaa Fatna Brahimi

Karim Toumi

Led by:

Dr. Ahmed Chaouki Lokbani

Class of 2023-2024

Alaa Fatna Brahimi's Dedication

Thank God for giving me the strength, wisdom and perseverance necessary to complete this work.

To my mother and father,
whose love, unconditional support and sacrifices made it all possible. Your belief in my abilities has always inspired me to give my best.

To my family,
for their unwavering support and understanding during the most demanding periods of this work.

To my friends & colleague,
for their constant support and encouragement, even in the most difficult times.

To my special person,
Mr. D.M (May God have mercy on him),for their unwavering encouragement and their confidence in me. I cannot express how much their help meant to me.

To my teachers and supervisors, Mr. Chouki Okbani,
for their valuable advice, expertise and dedication throughout this project.

To Karim Toumi,
my partner and collaborator, for your precious help and collaboration during this research. Your perseverance and dedication were essential to the success of this project.

To me,
I don't want to forget to thank myself. I would like to express my gratitude to myself for the hard work that I put into this thesis and for believing in myself throughout this project.

To all those who believed in me and accompanied me on this adventure,
This thesis is dedicated to you.

Karim Toumi’s Dedication

I would like to express my deepest gratitude to all those who have contributed to the successful completion of this thesis.

First and foremost, I would like to thank my advisor, Mr. Chouki Okbani, for their invaluable guidance, insightful suggestions, and continuous support throughout the duration of my research. Their expertise and dedication were instrumental in shaping the direction and outcome of this work.

I am also grateful to my colleagues and friends for their encouragement and constructive feedback, which greatly enriched my research experience.

A special thanks to my research partner, Alaa Brahimi, for their collaboration, hard work, and shared insights throughout this journey. Your dedication and teamwork were crucial to the success of our research.

To my family, thank you for your unwavering support and encouragement, which has been a constant source of motivation for me.

Finally, I would like to acknowledge myself for the perseverance, dedication, and hard work that I put into this thesis. This achievement would not have been possible without my own commitment and determination.

Thank you all.

Acknowledgements

Above all, we thank Almighty Allah for having granted us the strength and means to be able to carry out this work. It is with great pleasure that I express in these few lines my sincere gratitude and my love to all people that I know and to all those who contributed to my success.

We would like to express our deepest gratitude to all those who supported and guided us throughout the development of our thesis entitled "Comparative study of cryptography algorithms". Your expertise, wise advice and patience were essential to the successful completion of this work.

First of all, we would like to express our sincere gratitude to Mr. Ahmed Chaouki Lokbani for his advice, support, seriousness and compassion. In particular, we sincerely thank him for the time he devoted to guiding us, proofreading our work and providing us with constructive feedback. Your encouragement and availability were invaluable and contributed greatly to the quality and rigor of this research.

In addition, we would like to underline the honor that the members of the jury did us by agreeing to evaluate our work.

Your confidence in our abilities has given us the strength and motivation to overcome challenges and persevere in our efforts. Thanks to you, we were able not only to complete this ambitious project, but also to acquire skills and knowledge that will be invaluable to us for the rest of our academic and professional careers.

ملخص

التشفير هو علم وفن تأمين المعلومات عن طريق تحويلها لجعلها غير قابلة للقراءة لأي شخص غير مصرح له. وهو نقطة حاسمة في الأمن، وتحديداً أمن تكنولوجيا المعلومات، حيث يستخدم العديد من الخوارزميات لتشفير المعلومات وفك تشفيرها وتأمينها. تهدف الأطروحة إلى تحليل ومقارنة خوارزميات التشفير المختلفة للمساعدة في اتخاذ قرارات مستنيرة لأغراض محددة. كما تتضمن الدراسة تطوير موقع إلكتروني لمقارنة نتائج التشفير وفك التشفير بين الخوارزميات.

الكلمات المفتاحية: التشفير، المقارنة، الخوارزمية.

Abstract

Cryptography is the science and art of securing information by transforming it to make it unreadable.to any unauthorized person. A crucial point in security, specifically IT security, it uses several algorithms for encrypting, decrypting, and securing information. The thesis aims to analyze and compare various cryptographic algorithms to help make informed decisions for specific purposes. The study also includes the development of a website for comparing encryption and decryption results between algorithms.

Keywords: encryption, algorithmic, comparison.

Résume

La cryptographie est la science et l'art de sécuriser une information en la transformant pour la rendre illisible à toute personne non autorisée. Point crucial de la sécurité, notamment informatique, elle utilise plusieurs algorithmes pour chiffrer, décrypter et sécuriser l'information. La thèse vise à analyser et comparer divers algorithmes cryptographiques pour aider à prendre des décisions éclairées à des fins spécifiques. L'étude comprend également le développement d'un site Web permettant de comparer les résultats de cryptage et de décryptage entre les algorithmes.

Mots-clés : chiffrement, comparaison, algorithme.

Abbreviation list

AES: Advanced Encryption Standard.
CBC: Cipher Block Chaining.
CFB: Cipher Feedback.
DES: Data Encryption Standard.
ECB: Electronic Codebook.
RSA: Rivest–Shamir–Adleman.
IDS: Intrusion Detection Systems.
IPS: Intrusion Prevention Systems.
IV: initialization vector.
MD5: Message-Digest Algorithm5.
NIST: National Institute of Standards and Technology.
OFB: Output Feedback.
SHA-3: Secure Hash Algorithm Version 3.
SHA-256: Secure Hash Algorithm Version 256.
VPN: Virtual Private Network.

List of Figures

Figure 1.2-1: The scytale [3] 9
Figure 2.3-1:Caesar Shift [1] 10
Figure 3.2-1:ECB mode [3] 30
Figure 3.2-2:CBC mode [3] 30
Figure 3.3-4:Structure générale de l'AES [3] 32
Figure 3.3-5: DES encryptions process [13] 34
Figure 3.3-7: A diagram of a 3DES encryption and decryption system 35
Figure 3.4-1: The asymmetric crypto system. [3] 35
Figure 3.4-2:Adi Shamir, Ronald Rivest and Leonard Adleman [16] ..... Erreur ! Signet non défini.
Figure 3.4-3: RSA key for Encryption and decryption. 36
Figure 3.5-1Hash function. [[3] 37
Figure 3.5-2:Integrity check [3] 38
Figure 3.5-3:One SHA-3 Iteration [24] 39
Figure 3.5-5:MD5 algorithm [20] 42
Figure 3.5-6:Padding bits [20] 43
Figure 3.5-7:Input data to be hashed [20] 43
Figure 3.5-8:Data Flow with Intermediate States [20] 44
Figure 4.2-1:Use Case Diagram of comparison between cryptography algorithms 48
Figure 4.2-2:sequence Diagram of classical comparison 49
Figure 4.3-1:Homepage of the proposed system 53
Figure 4.3-2:Cryptography system category page 53
Figure 4.3-3:Symmetric Cryptography,3DES algorithm 54
Figure 4.3-4:Symmetric algo DES 54
Figure 4.3-5:symmetric algorithms comparison 55
Figure 5.4-1:Histogram explain comparison between Playfair & Vigenere using short Key & Medium text 59
Figure 5.4-2:Histogram explain comparison between Playfair & Vigenere using short Key & very long text ... 61
Figure 5.5-1:histogram explain comparison between AES, DES, 3DES using short Key and short text 64
Figure 5.5-2:Comparison between AES, DES,3DES in « ECB mode », using short text 66
Figure 5.5-3:histogram explain comparison between AES, DES, 3DES using Long text in ECB mode 67
Figure 5.5-4:Histogram explain comparison between AES, DES,3DES in CBC mode using medium text 68
Figure 5.5-5:Histogram explain Comparison between AES, DES,3DES in CBC mode using very long text 69
Figure 5.5-6:Histogram explain Comparison between AES key size in ECB mode using medium text 70
Figure 5.5-7:Histogram explain Comparison between AES key size in ECB mode using very long text 71
Figure 5.5-8:Histogram explain Comparison between AES key size in CBC mode using medium text 73
Figure 5.5-9:Histogram explain comparison between AES key size in CBC mode using long text 74
Figure 5.6-1:Histogram explain comparison between the algorithm AES and RSA using medium text. 76
Figure 5.6-2:Histogram explain comparison between algorithms AES and RSA using txt size 501 bits 78
Figure 5.7-1;Histogram explain comparison between RSA key size in « ECB mode » using text 5 80
Figure 5.8-1: Histogram explain comparison between hashing algorithms using medium text 82
Figure 5.8-2:Histogram explain Comparison between Hashing algorithms using very long Key using very long text 83

List of Table

Table 2.4-1:Example of Playfair encryption process(letters are in the same column) 13
Table 2.4-2:Example of Playfair encryption process (letters are in the same row) 13
Table 2.4-3:Example of Playfair encryption process(letters are neither of the above rules is true) 14
Table 3.6-1: SHA-3 Hashing Examples 40
Table 3.6-3: SHA-256 Hashing Examples 42
Table 3.6-4: MD5 Hashing Examples 42
Table 3.3-1:The characteristics of the material used 52
Table 5.4-2: Comparison between Playfair & Vigenère using short Key & medium text 59
Table 5.4-3: Comparison between Playfair & Vigenere using short Key and very long text 60
Table 5.4-5: Execution time of encryption and decryption of playfair and vignere for different text sizes 62
Table 5.5-1:Table of keys chosen from symmetric algos 63
Table 5.5-4:Comparison between symmetric algorithms (AES, DES,3DES) using long text 65
Table 5.5-5: keys chosen from symmetric algos mode ECB 66
Table 5.5-6: Comparison between AES, DES,3DES in « ECB mode », using short text 66
Table 5.5-8:Comparison between AES, DES,3DES in « ECB mode », using long text 67
Table 5.5-10: keys chosen from symmetric algos in CBC mode 67
Table 5.5-12:Comparison between AES, DES,3DES in « CBC mode », using medium text 68
Table 5.5-14:Comparison between AES, DES,3DES in « CBC mode » using Very long text 69
Table 5.5-15: keys chosen from AES algo in ECB mode 70
Table 5.5-16: Comparison between AES key size in « ECB mode » using medium text 70
Table 5.5-17:Comparison between AES key size in « ECB mode » 71
Table 5.5-19:Comparison between AES key size in « CBC mode » 72
Table 5.5-20:Comparison between AES key size in « CBC mode » 73
Table 5.6-1: keys chosen from AES & RSA algos 75
Table 5.6-2:Comparison between the algorithm AES and RSA using medium text. 76
Table 5.6-3: keys chosen from AES & RSA algos 77
Table 5.6-4:Comparison between algorithms AES and RSA using txt size 501 bits 78
Table 5.7-1:Comparison between RSA key size in « ECB mode » using text 5 80
Table 5.8-1:Comparison between hashing algorithms using medium text 82
Table 5.8-2:Comparison between Hashing algorithms using very long text 83

Glossary

In this presentation I will endeavor to always use the same terms. This paragraph presents the terms used as well as their meaning.

- Algorithm: set of instructions for performing a task.
- Ciphertext: This is the result of encrypting the plaintext.
- Cryptography: this branch brings together all the methods which make it possible to encrypt and decrypt a plain text in order to make it incomprehensible to anyone who is not in possession of the key to use to decipher it.
- Cryptanalysis: it is the art of revealing plain texts that have been encrypted without knowing the key used to encrypt the plain text.
- Cryptology: this is the science that studies secret communications. It is made up of two complementary areas of study: cryptography and cryptanalysis.
- Decryption is the method or algorithm used to transform ciphertext into plaintext.
- Decrypt: it is the action of finding the plain text corresponding to an encrypted text without having the key which was used for encryption. This word should therefore only be used in the context of cryptanalysis.
- Encryption: This is the method or algorithm used to transform plain text into cipher text.
- Encrypt: by rereading the definition of the word decrypt, we can realize that the word encrypt has no meaning and that its use should be forgotten. The word encryption doesn't make any more sense either.
- Encode, decode: it is a method or algorithm allowing you to modify the formatting of a message without introducing a secret element. Morse code is therefore a code since it transforms letters into lines and dots without any notion of secrecy. ASCII is also a code since it allows you to transform a letter into a binary value.
- Hash function: algorithm that transforms data into a unique digital fingerprint.
- Key: This is the shared secret used to encrypt plaintext into ciphertext and to decrypt ciphertext into plaintext. We can perfectly design an algorithm which does not use a key, in this case it is the algorithm itself which constitutes the key, and its principle must therefore in no case be revealed.
- Plain text: this is the message to be protected.

Table of Contents

1 Classical Cryptography 7
1.1 Introduction: 8
1.2 Definition of Cryptography: [1] 8
1.3 Classical cryptography 9
1.3.1 Monoalphabetic ciphers: 9
1.3.1.1 Caesar Cipher: 9
1.3.1.1.1 Working of Caesar: 10
1.3.2 Polyalphabetic: 10
1.3.2.1 Vigenère Cipher:[9] 11
1.3.2.2 The Playfair cipher: 12
1.3.2.2.1 The Playfair process cipher: 13
1.3.2.2.2 Rules for Encryption: 13
1.4 Conclusion 16
2 Modern cryptography 28
2.1 introduction: 29
2.2 modern encryption families: 29
2.2.1 Code by Blocks: 29
2.2.1.1 ECB Block Cipher Modes: 29
2.2.1.2 CBC Block Chiper Modes: [3] 30
2.3 Symmetric Cryptography: 31
2.3.1 Symmetric cryptography algorithms: 31
2.3.1.1 AES (advanced encryption standard): 31
2.3.1.1.1 Encryption: 31
2.3.1.1.2 Decryption: [4] 32
2.3.1.2 DES (Data Encryption Standard): 32
2.3.1.2.1 Working of the cipher: [18] 33
2.3.1.3 3DES (Data Encryption Standard): 34
2.3.1.3.1 Encryption Process: 34

TABLE OF CONTENTS

2.3.1.3.2 Decryption Process: 35
2.4 Asymmetric Cryptography: 35
2.4.1 RSA (Rivest-Shamir-Adleman): 36
2.4.1.1.1 How RSA works:[14] 36
2.5 Hash function: 37
2.5.1 Principle: [3] 38
2.5.2 Hashing algorithms: 38
2.5.2.1 SHA-3: 38
2.5.2.1.1 How it works: [24] 39
2.5.2.2 SHA-256: 41
2.5.2.2.1 SHA-256 Algorithm :23 41
2.5.2.3 MD5: [20] 42
2.5.2.3.1 How MD5 works: [20] 43
2.6 Conclusion: 44
3 Conception &Architecture of system 46
3.1 Introduction: 47
3.2 Conception: 47
3.2.1 Problem: 47
3.2.2 Objectif: 47
3.2.3 Overall operation: 47
3.2.3.1 Use case diagram 48
3.2.3.2 Sequence Diagram: 48
3.3 Architecture of system: 50
3.3.1 Framework: 50
3.3.2 Software environment: 50
3.3.3 Language: 50
3.3.4 Another tool: 51
3.3.5 Hardware environment: 52
3.3.6 Process of system: 52
3.3.6.1 Homepage: 52
3.3.6.2 Cryptography system category page: 53
3.3.6.3 Symmetric Cryptography: 53
3.3.6.4 Algorithm category: 54
3.4 Conclusion: 55
4 Experimentation and evaluation of results 56

TABLE OF CONTENTS

4.1 Introduction: 57
4.2 Cryptography algorithm comparison: 57
4.3 Comparison Criteria: 58
4.4 Comparison between classical algorithms: 58
4.4.1 Comparison between Playfair & Vigenère using medium text: 58
4.4.1.1 Observation: 59
4.4.1.2 Conclusion: 60
4.4.2 Comparison between Playfair & Vigenère using very long text 60
4.4.2.1 Observation: 61
4.4.2.2 Explanation of the differences: 61
4.4.2.3 Conclusion: 62
4.5 Comparison between symmetric algorithms: 62
4.5.1 Comparison between symmetric algorithms AES<128: 63
4.5.1.1 Observations: 64
4.5.1.2 Conclusion: 64
4.5.1.3 General conclusion: 65
4.5.2 Comparison between symmetric algorithms Mode ECB: 65
4.5.2.1 Conclusion: 67
4.5.3 Comparison between symmetric algorithms mode CBC: 67
4.5.3.1 Discussion: 69
4.5.3.2 Conclusion: 69
4.5.4 Comparison between AES algorithm key size: 70
4.5.4.1 Comparison between AES key size in ECB mode: 70
4.5.4.1.1 Observations: 71
4.5.4.1.2 Conclusion: 72
4.5.4.2 Comparison between AES key size in CBC mode: 72
4.5.4.2.1 Conclusion 74
4.5.4.3 General conclusion for symmetric algorithms: 74
4.5.4.4 Security vs. Performance Trade-off: 75
4.5.4.5 Choosing the Right Algorithm and Mode: 75
4.6 Comparison between AES & RSA algorithms: 75
4.6.1 Comparison between AES and RSA using medium text: 76
4.6.1.1 Discussion: 76
4.6.1.2 Conclusion: 77
4.6.2 Comparison between AES and RSA using text size 501 bits: 77

4.6.2.1 Conclusion: 78
4.7 RSA key size comparison: 78
4.7.1 Key Size and Maximum Input Length: 79
4.7.2 Maximum Input Length example calculations: 79
4.7.3 Discussion: 80
4.7.4 Conclusion: 81
4.8 Hache Algorithm Comparison: 81
4.8.1 Comparison between Hashing algorithms using medium text: 81
4.8.1.1 Observations 82
4.8.1.2 Conclusion 83
4.8.2 Comparison between Hashing algorithms using very long text: 83
4.8.2.1 Observation: 84
4.8.2.2 Conclusion: 84
4.8.3 Hash tables analyses: 84
4.9 Conclusion: 85
Abbreviation list **Erreur ! Signet non défini.**
List of Figures **Erreur ! Signet non défini.**
List of Table **Erreur ! Signet non défini.**
Glossary **Erreur ! Signet non défini.**
Bibliography 87

General Introduction

Computer security (information technology or IT security) is the process of protecting an organization's IT (computer systems, networks, digital devices, and data) from unauthorized access, data breaches, cyberattacks and other malicious activities.
IT security is broad and often includes a combination of technologies and security solutions combined to address vulnerabilities. Cryptography is indeed an essential tool for computer security. This is done by making data unreadable by unauthorized people, it is used in many areas of computer security, including protecting communications, authenticating users, verifying data integrity, and securing online transactions.

So, we can say that cryptography is the science of protecting sensitive information from disclosure, alteration or falsification. It uses mathematical techniques to transform readable data (plain text) into an intelligible format (ciphertext). Only someone with the appropriate decryption key can recover the original message.

Generally, cryptography is a writing technique that uses a secret code or encryption key to write an encrypted message, depending on mathematical techniques to transform readable data (plain text) into an intelligible format (ciphertext). There are many cryptographic algorithms that can be used to encrypt messages and allow recipients to decrypt them. Some can be considered simple (e.g. moving letters of the alphabet to the right or left for a certain number of notes), while others provide an almost absolute level of security.

In today's changing landscape and with the rise of cyber threats it's crucial to delve into and contrast various cryptography algorithms to safeguard sensitive information. Despite the abundance of algorithms selecting the suitable one, for specific purposes can pose a challenge. Each algorithm brings its set of pros and cons in terms of performance, security, energy usage and complexity. Having insights is key to making informed decisions when choosing the right algorithm for each unique scenario. This thesis sets out to offer a analysis of prevalent cryptography algorithms in use today.

The primary goals of this study are to furnish comprehensive information that aids decision making regarding cryptography algorithm selection and sheds light on the strengths and weaknesses in each algorithm to foster a deeper understanding of their capabilities and limitations. To meet these objectives this thesis delves into following questions: What're the key advantages and drawbacks associated with cryptography

algorithms? How does the performance of major cryptographic algorithms compare in terms of encryption/decryption time and resource consumption? Which cryptography algorithms provide the best flexibility and adaptability for use in varied scenarios.

In our culminating project, we aim to explore a comparison of algorithms by developing a website that enables users to compare encryption or decryption outcomes across algorithms along, with their metrics. The dissertation is organized in the following manner:

- The first chapter presents classical cryptography, its algorithms and how work each algorithm is used.
- The modern cryptography system is presented in the second chapter.
- The third chapter contains the conception and architecture of our website.
- while its realization and comparison are covered in the fourth, as well as a detail of the key points cited in each work treated.

We conclude our thesis with a broad statement

CHAPTER 1

1 Classical Cryptography

1.1 Introduction:

Cryptography has evolved over the centuries, from simple methods such as letter substitution to complex algorithms based on advanced mathematical principles. Its fundamental role is to secure electronic communications and transactions in an increasingly interconnected world.

The history of encryption traces a fascinating epic in which cryptographers ("crypters") and cryptanalysts ("decryptors") engage in a fierce battle, an eternal restart of the development of an algorithm by some, of decoding by others, of development a new, more powerful algorithm, etc.[01]

1.2 Definition of Cryptography: [1]

Cryptography is the set of techniques for securing communications and data by transforming information in such a way as to make it incomprehensible to any unauthorized person. It is based on the use of mathematical and computer algorithms to encrypt and decrypt data, verify the authenticity of communicating parties, ensure the integrity of information, and guarantee the non-repudiation of transactions.

Cryptography, or the art of hiding and protecting information, has a long history dating back to ancient times. The first traces of cryptographic techniques date back to ancient Egypt, where hieroglyphs were carved in a way that rendered difficult reading. The oldest known ciphers are presented under the form of hieroglyphs found on monuments dating from almost 3,000 years BC. For a long time, hieroglyphs were considered indecipherable, before the discovery of the famous Rosetta Stone and the work of Jean-François Champollion made it possible to unravel its mysteries. But let's go back to Antiquity, to the 6th century before our era, in the Greek city of Sparta and its famous scytale, a thick stick around which the sender wound a strip of parchment to write your message. Only the parchment was then sent to the recipient. If this person had a stick of the same diameter, she could then wind the strip around it in order to decrypt the message. Encryption methods of this type – which involve changing the order of letters – fall into the category of “transposition encryption”.
The Spartans used the scytale, a cylinder around which a strip of parchment was wrapped to write secret messages. This is what the figure 1.2-1 show.

Figure 1.2-1: The scytale [3]

1.3 Classical cryptography

Classical cryptography refers to cryptographic methods that were widely used it based on complex mathematical algorithms. These methods were often based on substitutions, transpositions and other simple techniques.

Substitution: is a technique in which each element of plain text, whether letters, numbers or symbols, is replaced by another element according to a specific rule. There are several substitution methods, each of which has its own characteristics.
There are three types of substitutions:

- monoalphabetic: replaces each letter of the message with another letter of the alphabet;
- polyalphabetic: uses a sequence of monoalphabetic numbers (the key) reused periodically;
- polygrams: replaces a group of characters in the message with one another group of characters.

Transposition: Transposition methods involve rearranging letters or symbols in a message without changing them.

1.3.1 Monoalphabetic ciphers:

Monoalphabetic substitution ciphers represent a foundational type of encryption where each letter in a plaintext message is systematically replaced with a single alternative letter or symbol, creating the ciphertext. This substitution remains consistent throughout the encryption process. Imagine a secret codebook where each letter in the alphabet has a corresponding code assigned to it. Encryption involves replacing each plaintext letter with its designated code from the codebook.[8]

1.3.1.1 Caesar Cipher:

Caeser cipher is a type of monoalphabetic cipher where every character of plain text is mapped to another character by a distance of 3. It is essentially a type of additive cipher where the key value is always 3. As it seen in the figure1.3-1 .

For example, if the plain text has a character 'a' then the value of its cipher text counterpart will be 'd' since the value of the key is 3 in the Caeser cipher.[5]

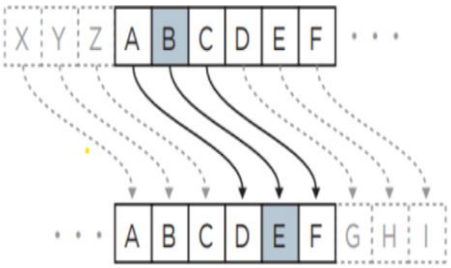


Figure 1.3-1:Caesar Shift [1]

1.3.1.1.1 Working of Caesar:

The mathematical representation of the Caeser cipher is:

- Encryption process:

The formula for encryption of plain text to cipher text in Caeser cipher is
C=(P+3) mod 26.
Here, P is the plain text, C is the cipher text and 3 is the key because it always remains the same in the Caeser cipher.[5]

- Decryption process:

The formula for the decryption process of cipher text to plain text in Caeser cipher is
P=(C−3) mod 26.
Here, P and C are plain text and cipher text respectively, and 3 is the key.[5]

1.3.2 Polyalphabetic:

A polyalphabetic cipher is a cryptographic method that uses multiple substitution alphabets to encode a message. Unlike monoalphabetic ciphers, where each letter is consistently replaced by the same letter throughout the message, this means that the same letter in the plaintext can be encrypted to different letters in the ciphertext, depending on its position in the message. polyalphabetic ciphers employ different substitution rules for different parts of the message. This introduces a layer of complexity that enhances the security of the encoded information.

One of the most well-known polyalphabetic ciphers is the Vigenère cipher, named for the 16th-century French cryptographer Blaise de Vigenère. In the Vigenère cipher, a keyword is repeated to generate a series of different Caesar ciphers. Each letter in the plaintext is then shifted according to the corresponding letter in the keyword.

In the simplest systems of the Vigenère type, the key is a word or phrase that is repeated as many times as required to encipher a message.

1.3.2.1 Vigenère Cipher:[9]

Vigenère Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. The encryption of the original text is done using the Vigenère square or Vigenère table.
Vigenère Cipher is a polyalphabetic encryption which is based on:

- The same letter will be encrypted in different ways
- Process one digraph (sequence of letters) at a time

This encryption uses a key that defines the offset for each letter of the message (A: 0 notch offset, B: 1 notch, C: 2 notches, ..., Z: 25 notches)

Encryption: $c_i = E_{k_i \bmod d}(m_i) \bmod 26 = m_i + k_{i \bmod d} \bmod 26$

Decryption: $M_i = D_{k_i \bmod d}(c_i) = c_i - k_{i \bmod d} \bmod 26$

Key: $K = (k_1, k_2, \ldots, k_d)$

1.3.2.2 Vigenère algorithm:

```
function vigenereEncrypt(plaintext, key):
begin
// Convert plaintext and key to uppercase
plaintext = uppercase(plaintext)
key = uppercase(key)
   // Initialize variables
encryptedText = ""
keyIndex = 0
   // Loop through each character in the plaintext
for each character in plaintext:
   // Calculate the offset for the current character of the plaintext
   offset = (charToNumber(character) + charToNumber(key[keyIndex])) mod 26
         // Convert the offset back to a character and append it to the encrypted text
   encryptedText += numberToChar(offset)
         // Move to the next character in the key (looping if necessary)
   keyIndex = (keyIndex + 1) mod length(key)
return encryptedText
end
function vigenereDecrypt(ciphertext, key):
begin
// Convert ciphertext and key to uppercase
ciphertext = uppercase(ciphertext)
key = uppercase(key)
// Initialize variables
decryptedText = ""
```

```
    keyIndex = 0

    // Loop through each character in the ciphertext
    for each character in ciphertext:
        // Calculate the offset for the current character of the ciphertext
        offset = (charToNumber(character) - charToNumber(key[keyIndex]) + 26) mod 26

        // Convert the offset back to a character and append it to the decrypted text
        decryptedText += numberToChar(offset)

        // Move to the next character in the key (looping if necessary)
        keyIndex = (keyIndex + 1) mod length(key)

    return decryptedText
    end
function charToNumber(character):
    begin
    // Convert character to its corresponding number (0-25)
    return ascii(character) - ascii('A')
  end
function numberToChar(number):
    begin
    // Convert number to its corresponding character (A-Z)
    return char((number + 26) mod 26 + ascii('A'))
    end
main()
begin
    write("Enter the plaintext: ")
    plaintext = read()
    write("Enter the key: ")
    key = read()
    // Encryption
    ciphertext = vigenereEncrypt(plaintext, key)
    write("Encrypted text: ", ciphertext)
    // Decryption
    decryptedText = vigenereDecrypt(ciphertext, key)
    write("Decrypted text: ", decryptedText)
    end
```

1.3.2.3 The Playfair cipher:

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who

promoted the use of the cipher. In playfair cipher unlike traditional cipher, we encrypt a pair of alphabets(digraphs) instead of a single alphabet.
It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment.[12]

1.3.2.3.1 The Playfair process cipher:

The Algorithm consists of 2 steps:[12]

- Generate the key Square(5×5):
- The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.
- The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.
- Algorithm to encrypt the plain text: The plaintext is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter.

1.3.2.3.2 Rules for Encryption:

If both the letters are in the same column: Take the letter below each one (going back to the top if the bottom). As it seen in the table 1.3-1.
For example:

Diagraph: “me”
Encrypted Text: cl => m -> c
e -> l

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Table 1.3-1:Example of Playfair encryption process(letters are in the same column)

If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position). As it seen in the table 1.3-2.

For example:
Diagraph: “st”
Encrypted Text: tl =>s -> t
t -> l

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Table 1.3-2:Example of Playfair encryption process (letters are in the same row)

If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle. As it seen in the table 1.3-3

For example:.

Diagraph: “nt”
Encrypted Text: rq => n -> r
t -> q

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Table 1.3-3:Example of Playfair encryption process(letters are neither of the above rules is true)

1.3.2.4 Playfair algorithm:

```
function generateKeySquare(key):
    begin
    key = removeDuplicates(key)
    keySquare = 5x5 grid
    alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
    keyIndex = 0
    for each character in key:
        add character to keySquare
        remove character from alphabet
    for each character in alphabet:
        add character to keySquare
    return keySquare

    end
function removeDuplicates(key):
    begin
    uniqueKey = ""
    for each character in key:
        if character not in uniqueKey:
            add character to uniqueKey
    return uniqueKey
    end
function encrypt(plaintext, keySquare):
    begin
    plaintext = preprocess(plaintext)
    ciphertext = ""
    for each digraph in plaintext:
        if digraph[0] == digraph[1]:
            digraph[1] = 'Z'
        row1, col1 = getCoordinates(digraph[0], keySquare)
        row2, col2 = getCoordinates(digraph[1], keySquare)
        if row1 == row2:
            ciphertext += keySquare[row1][(col1 + 1) % 5]
            ciphertext += keySquare[row2][(col2 + 1) % 5]
        else if col1 == col2:
            ciphertext += keySquare[(row1 + 1) % 5][col1]
```

```
            ciphertext += keySquare[(row2 + 1) % 5][col2]
        else:
            ciphertext += keySquare[row1][col2]
            ciphertext += keySquare[row2][col1]
    return ciphertext
    end
function decrypt(ciphertext, keySquare):
    begin
    plaintext = ""
    for each digraph in ciphertext:
        row1, col1 = getCoordinates(digraph[0], keySquare)
        row2, col2 = getCoordinates(digraph[1], keySquare)
        if row1 == row2:
            plaintext += keySquare[row1][(col1 - 1 + 5) % 5]
            plaintext += keySquare[row2][(col2 - 1 + 5) % 5]
        else if col1 == col2:
            plaintext += keySquare[(row1 - 1 + 5) % 5][col1]
            plaintext += keySquare[(row2 - 1 + 5) % 5][col2]
        else:
            plaintext += keySquare[row1][col2]
            plaintext += keySquare[row2][col1]

return plaintext
    end
function preprocess(text):
    begin
    text = text.replace("J", "I")
    text = text.replace(" ", "") // Remove spaces
    text = text.upper() // Convert to uppercase
    // Add 'Z' to last letter if the length is odd
    if length(text) % 2 != 0:
        text += 'Z'
    return text
    end
function getCoordinates(character, keySquare):
    begin
    for each row in keySquare:
        for each col, value in enumerate(row):
            if value == character:
                return row, col
    end
main()
begin
    write("Enter the plaintext: ")
    plaintext = read()
```

```
write("Enter the key: ")
key = read()

keySquare = generateKeySquare(key)


write("Plaintext: ", plaintext)
ciphertext = encrypt(plaintext, keySquare)
write("Ciphertext: ", ciphertext)
write("Ciphertext: ", ciphertext)
decryptedText = decrypt(ciphertext, keySquare)
write("Decrypted text: ", decryptedText)
end
```

1.4 Conclusion

In the evolution of communication security, classical cryptography played a pivotal role by employing methods like the Caesar cipher, substitution, and transposition. Despite its historical significance, classical cryptography is now deemed obsolete, primarily due to its substantial vulnerabilities, including susceptibility to frequency analysis and a deficiency in secure key implementation. That is what led to the emergence of modern cryptography, which is based on solid mathematical foundations and sophisticated algorithms, which we will explore in detail in the next chapter.

CHAPTER 2

2 Modern cryptography

2.1 introduction:

Modern cryptography comprises a wide range of techniques and algorithms used to secure communications, protect data, and ensure the authenticity of information in various contexts, such as online communications, financial transactions, computer networks, and the security of private life. In other words, it aims to make the information confidential, guarantee its authenticity, and ensure that it has not been modified. It relies on mathematical principles and complex algorithms to encrypt and decrypt data.
In the previous chapter, we discussed classical cryptography. In this chapter, we will explain modern cryptography and its algorithms. We will focus on symmetric and asymmetric cryptography, as well as hashing, detailing how they work and the functioning of each algorithm.

2.2 modern encryption families:

Stream codes and block codes are two fundamental approaches to modern cryptography. This approach is commonly used in secure communications over the Internet, encrypted data storage, and other applications requiring a high level of security. We will particularly focus on block encryption modes, highlighting two specific modes: Electronic Codebook (ECB), Cipher Block Chaining (CBC).

2.2.1 Code by Blocks:

In modern cryptography, block ciphers are algorithms used to secure data by dividing it into blocks and repeatedly encrypting them. The division of data is into fixed size blocks before encrypting them.
Each block of data is then treated as an independent unit during the encryption process. This approach is commonly used in secure Internet communications, encrypted data storage, and other applications requiring a high level of security. Block cipher algorithms can work on blocks of various sizes, but they are often designed to work with 64- or 128-bit blocks. Block cipher modes, such as ECB, CBC, CFB, and OFB, are used to specify how blocks of data are processed and encrypted in these systems.[3]

2.2.1.1 ECB Block Cipher Modes:

ECB (Electronic Code Book) mode is the simplest mode of use, where each block of plain text is encrypted independently. However, this mode is considered less secure, as it allows an attacker to infer information about the original message by analyzing the encrypted blocks. This figure explains the ECB encryption process. [3]

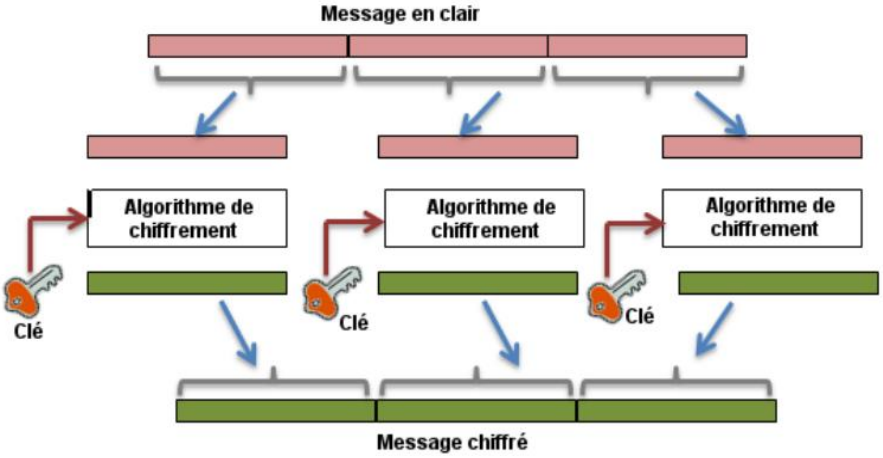


Figure 2.2-1:ECB mode [3]

2.2.1.2 CBC Block Chiper Modes: [3]

Cipher Block Chaining (CBC) mode is more secure than ECB mode because it uses the ciphertext of the previous block to encrypt the current block. This means that each block of plaintext is encrypted using a different key, making it more difficult for an attacker to infer information about the original message.
This mode uses an initialization vector (IV) combined with the previous block of ciphertext to encrypt each block of data.
Each block is first XORed with the previous encrypted block before being encrypted itself. This figure explains the CBC encryption process.

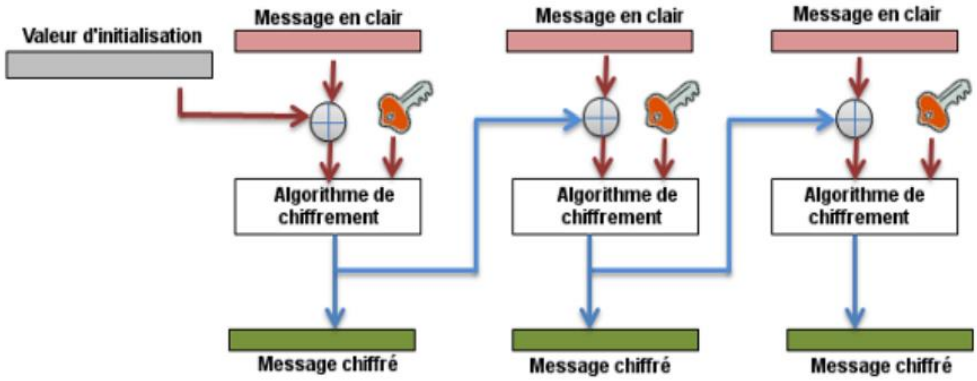


Figure 2.2-2:CBC mode [3]

The initialization vector (IV) :

The initialization vector (IV) is used to initialize the encryption process. Its main purpose is to disrupt recurring patterns in the plaintext that could be exploited by an attacker to discover encryption information.
In CBC mode, the IV is combined with the first plaintext block before encryption. For subsequent blocks, the previous cipher block is combined with the plaintext before being encrypted. This means that each block of plaintext is encrypted in a manner dependent on the IV and the previous block, which increases security. [3]

2.3 Symmetric Cryptography:

Symmetric cryptography, also called secret key cryptography, is an encryption method where the same key is used to encrypt and decrypt data. This method has been used since ancient times. It relies on the use of a shared key between communicating parties to encrypt and decrypt data. This key must be shared secretly between the sender and the recipient. Symmetric cryptography is one of the oldest forms of cryptography and is still widely used today due to its simplicity and effectiveness.[3]

2.3.1 Symmetric cryptography algorithms:

2.3.1.1 AES (advanced encryption standard):

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm widely used to secure sensitive data in a variety of applications, including Internet communications, data storage, and computer system security.
AES is a block cipher that works on 128-bit blocks, using 128-, 192-, or 256-bit keys. The number of iterations varies between 10, 12 and 14 where each round uses a key generated from the main key.
The encryption process consists of a series of rounds, with each round comprising four steps: byte substitution, row shifting, column shuffling, and adding a round key. These steps are designed to provide a high level of security and resistance to various types of attacks, such as differential and linear cryptanalysis.

2.3.1.1.1 Encryption:

The state or block in plain text is represented by a matrix. The number of rows in this matrix always equals 4 but the number of equal columns changes depending on the size of the block: 4 columns correspond to a block of 128 bits, 6 columns to a block of 192 bits, and 8 columns to a block of 256 bits.[3]
AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.

[b0 | b4 | b8 | b12 |
| b1 | b5 | b9 | b13 |
| b2 | b6 | b10| b14 |
| b3 | b7 | b11| b15]

Each round comprises of 4 steps:

- SubBytes
- ShiftRows
- MixColumns
- Add Round Key

The last round doesn't have the MixColumns round.
The SubBytes does the substitution and ShiftRows and MixColumns performs the permutation in the algorithm.
Here, the 16 bytes is not considered as a grid but just as 128 bits of data. [4]

The figure 2.3-1 explains the AES encryption process

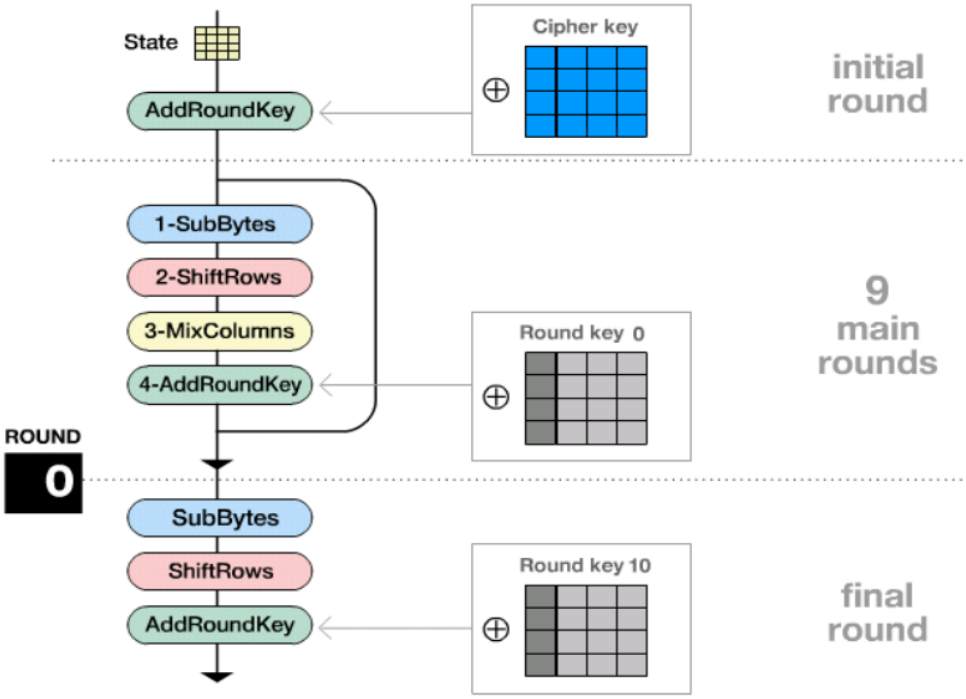


Figure 2.3-1:Structure générale de l'AES [3]

After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

2.3.1.1.2 Decryption: [4]

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes. Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round in decryption are as follows:

- Add round key
- Inverse MixColumns.
- ShiftRows.
- Inverse SubByte.

2.3.1.2 DES (Data Encryption Standard):

The DES (Data Encryption Standard) algorithm is a symmetric-key block cipher created in the early 1970s by an IBM team and adopted by the National Institute of Standards and Technology (NIST). The algorithm takes the plain text in 64-bit blocks and converts them into ciphertext using 48-bit keys.
Since it's a symmetric-key algorithm, it employs the same key in both encrypting and decrypting the data. If it were an asymmetrical algorithm, it would use different keys for encryption and decryption.

DES is based on the Feistel block cipher, called LUCIFER, developed in 1971 by IBM cryptography researcher Horst Feistel. DES uses 16 rounds of the Feistel structure, using a different key for each round.
DES became the approved federal encryption standard in November 1976 and was subsequently reaffirmed as the standard in 1983, 1988, and 1999. [18]

2.3.1.2.1 Working of the cipher: [18]

To put it in simple terms, DES takes 64-bit plain text and turns it into a 64-bit ciphertext. And since we're talking about a symmetric algorithm, the same key is used when it's time to decrypt the text. This is what the figure 2.3-2 show.

The algorithmic process breaks down into the following steps:

1. The process begins with the 64-bit plain text block getting handed over to an initial permutation (IP) function.
2. The initial permutation (IP) is then performed on the plain text.
3. Next, the initial permutation (IP) creates two halves of the permuted block, referred to as Left Plain Text (LPT) and Right Plain Text (RPT).
4. Each LPT and RPT goes through 16 rounds of the encryption process.
5. Finally, the LPT and RPT are rejoined, and a Final Permutation (FP) is performed on the newly combined block.
6. The result of this process produces the desired 64-bit ciphertext.

The encryption process step (step 4, above) is further broken down into five stages:

1. Key transformation
2. Expansion permutation
3. S-Box permutation
4. P-Box permutation
5. XOR and swap

For decryption, we use the same algorithm, and we reverse the order of the 16 round keys.

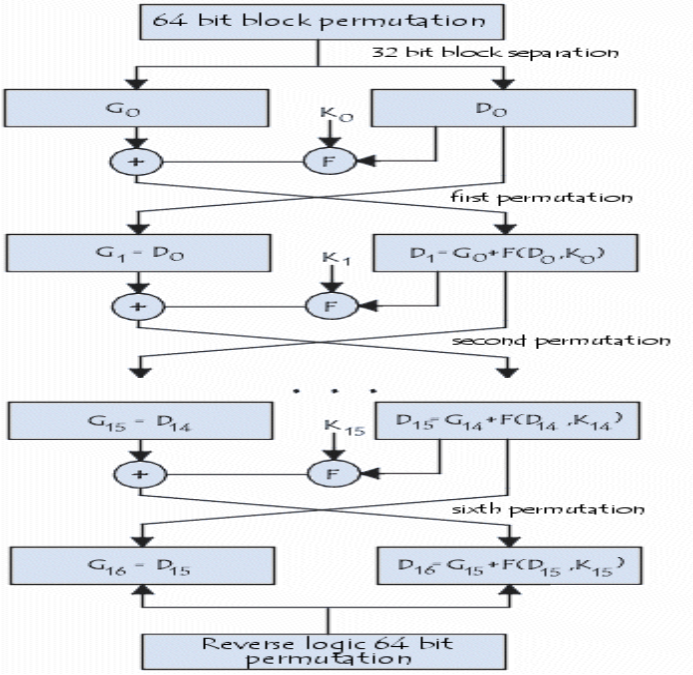


Figure 2.3-2: DES encryptions process [13]

2.3.1.3 3DES (Data Encryption Standard):

Although it's officially known as the Triple Data Encryption Algorithm (3DEA), it is most commonly referred to as 3DES. This is because the 3DES algorithm uses the Data Encryption Standard (DES) cipher three times to encrypt its data.

DES is a symmetric-key algorithm based on a Feistel network. As a symmetric key cipher, it uses the same key for both the encryption and decryption processes. The Feistel network makes both of these processes almost exactly the same, which results in an algorithm that is more efficient to implement.

DES has both a 64-bit block and key size, but in practice, the key only grants 56-bits of security. 3DES was developed as a more secure alternative because of DES's small key length. In 3DES, the DES algorithm is run through three times with three keys; however, it is only considered secure if three separate keys are used.[6]

2.3.1.3.1 Encryption Process:

The encryption process of 3DES involves the following steps:

Key Generation: Three unique keys are generated using a key derivation algorithm.

Initial Permutation: The 64-bit plaintext is subjected to an initial permutation.

Three Rounds of Encryption: The plaintext is encrypted three times, each time using a different key, to create three layers of encryption.

Final Permutation: After the three rounds of encryption, a final permutation is applied to the output to produce the ciphertext.[19]

2.3.1.3.2 Decryption Process:

The decryption process of 3DES is simply the reverse of the encryption process, with the ciphertext being fed into the algorithm and the steps being performed in reverse order, using the three keys in reverse order.[19]
As shown in the figure below.

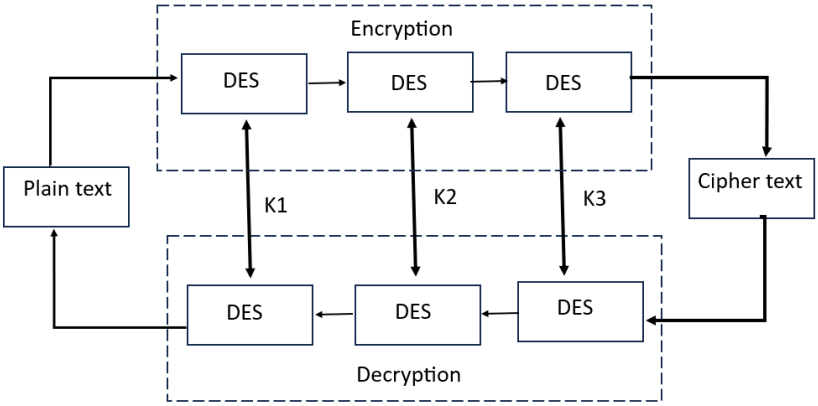


Figure 2.3-3: A diagram of a 3DES encryption and decryption system

2.4 Asymmetric Cryptography:

Asymmetric cryptography plays a crucial role in computer security. This innovation relies on the use of mathematically related but distinct key pairs: a public key for encryption and a private key for decryption. While symmetric cryptography relies on a single key for encryption and decryption
The public key can be freely shared with anyone, while the private key must remain confidential to its owner. The development of asymmetric cryptography aimed to solve the key exchange problem inherent in symmetric cryptography, where the same key is used for encryption and decryption. The figure below explains the asymmetric crypto system.

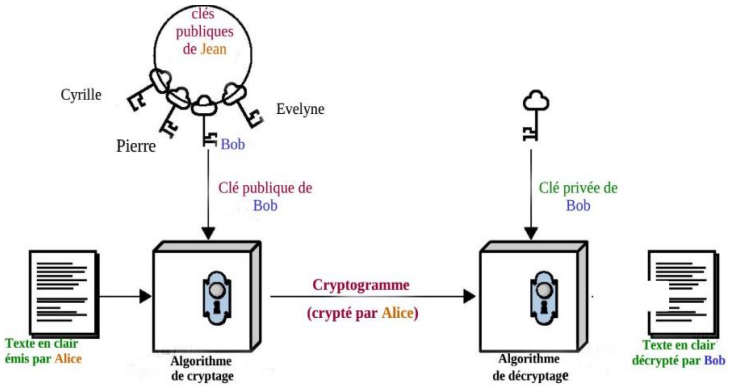


Figure 2.4-1: The asymmetric crypto system. [3]

2.4.1 RSA (Rivest-Shamir-Adleman):

The RSA algorithm was invented in 1977 by Ronald Rivest, Adi Shamir and Leonard Adleman. This is the first instance of public key cryptography methods, the principle of which was introduced as a concept by Whitfield Diffie and Martin Hellman. Currently, it is undoubtedly the best known and most used asymmetric cryptosystem in the world. The RSA algorithm is used for public key cryptography and is based on the fact that it is easy to multiply two large prime numbers but difficult to factor the product. This is the most common example of asymmetric cryptography, still considered secure, with current technology, for sufficiently large keys (1024, 2048 or even 4096 bits).[16]

The algorithm for generating the public and private keys is the most complex part of RSA encryption. Two large prime numbers, p and q, are generated using the Rabin-Miller primality test. A cipher modulus n is calculated by multiplying p and q. This number is used by both keys (public and private) and forms the link between them. Its length, usually expressed in bits, is called the key length. The public key consists of the modulus n and a public exponent, e, normally set to 65537, as this is not too large a prime number. The number e need not be a secretly chosen prime, as the public key is shared with everyone. The private key is made up of the modulus n and the private exponent d, which is calculated using Euclid's algorithm extended to give the modular inverse with respect to the value of the Euler index in n. [17]

2.4.1.1.1 How RSA works:[14]

Key Generation:
You need to generate public and private keys before running the functions to generate your ciphertext and plaintext. They use certain variables and parameters, all of which are explained below:
Choose two large prime numbers (p and q)
Calculate n = p*q and z = (p-1)(q-1)
Choose a number e where 1 < e < z
Calculate d = e-1mod(p-1)(q-1)
You can bundle private key pair as (n,d)
You can bundle public key pair as (n,e)
The figure below shown the RSA key for encryption and decryption.

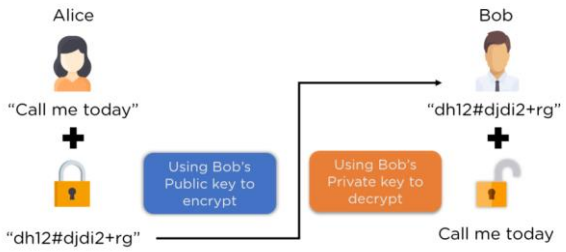


Figure 2.4-2: RSA key for Encryption and decryption. [14]

Encryption/Decryption Function:
Once you generate the keys, you pass the parameters to the functions that calculate your ciphertext and plaintext using the respective key.
If the plaintext is m, ciphertext = me mod n.
If the ciphertext is c, plaintext = cd mod n
To understand the above steps better, you can take an example where p = 17 and q=13. Value of e can be 5 as it satisfies the condition 1 < e < (p-1)(q-1).
N = p * q = 221
D = e-1mod(p-1)(q-1) = 29
Public Key pair = (221,5)
Private Key pair = (221,29)
If the plaintext(m) value is 10, you can encrypt it using the formula me mod n = 82.
To decrypt this ciphertext(c) back to original data, you must use the formula cd mod n = 29.

2.5 Hash function:

Hashing is a computer process that takes variable-size data as input and transforms it into a fixed-size value, usually in the form of a character string. This hash value is unique for a given data set and is generated deterministically by a hash function. as it seen in the figure 2.5-1.
A hash function is a method of characterizing information by reducing an input of any size to a fixed-size fingerprint.
It must uniquely associate a hash with plaintext, ensuring that any modification of the document will result in a modification of its hash.
The hash function must be one-way, making it impossible to recover the original message from the hash.

In cryptography, hash functions must meet stricter constraints to ensure the security of information exchanges.
Hashing is widely used to secure data, verify its integrity, facilitate efficient data searching, and other IT applications.

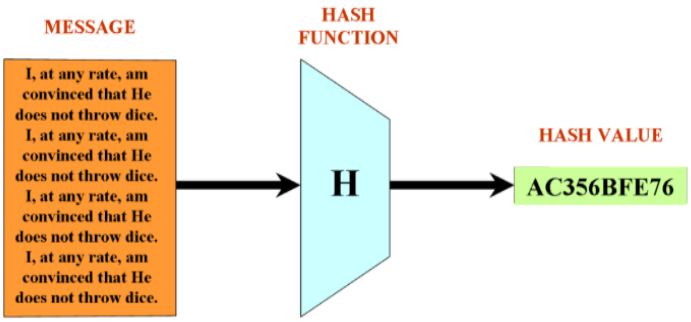


Figure 2.5-1Hash function. [[3]]

2.5.1 Principle: [3]

- A hash function calculates the fingerprint y of a message x. This function must be a one-way function.
- It must also be very sensitive so that a small modification of the message leads to a large modification of the fingerprint.
- By sending the message accompanied by its fingerprint, the recipient can ensure the integrity of the message by recalculating the summary upon arrival and comparing it to that received.
- If the two summaries are different, this means that the file is no longer the same as the original: it has been altered or modified by a third party. As shown in the figure below.

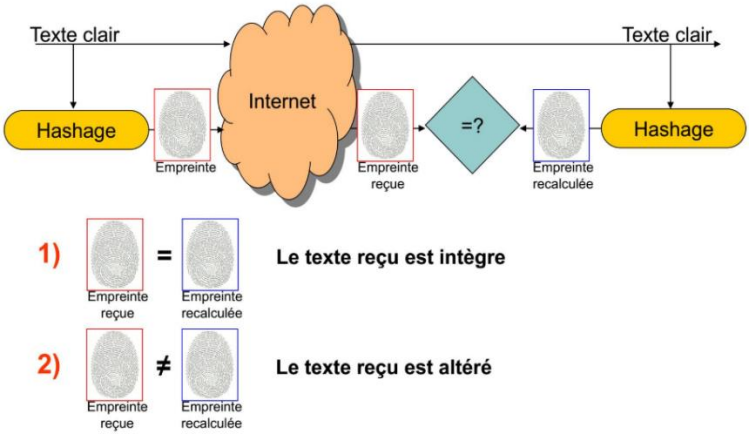


Figure 2.5-2:Integrity check [3]

2.5.2 Hashing algorithms:

Among the existing hash functions there is the range of functions proposed by NIST, the functions SHA-1 SHA-3 SHA-256 .... In this section, we will choose the MD5 algorithm, SHA-3 and SHA- 256 to explain its definition, characteristics and how it works.

2.5.2.1 SHA-3:

It is the latest standard released by NIST on August 5, 2015. SHA-3 is internally different from the MD5-like structure of SHA-1 and SHA-2. NIST does not currently plan to withdraw SHA-2 or remove it from the revised Secure Hash Standard. The purpose of SHA-3 is that it can be directly substituted for SHA-2 in current applications if necessary, and to significantly improve the robustness of NIST's overall hash algorithm toolkit. [22]

To ensure the message can be evenly divided into r-bit blocks, padding is required. SHA-3 uses the pattern 10*1 in its padding function: a 1 bit, followed by zero or more 0 bits (maximum r − 1) and a final 1 bit. The block transformation f, which is Keccak-f for SHA-3, is a permutation that uses XOR, AND and NOT operations, and is designed

for easy implementation in both software and hardware. In 2016 the same team that made the SHA-3 functions and the Keccak algorithm introduced faster reduced-rounds (reduced to 12 and 14 rounds, from the 24 in SHA-3) alternatives which can exploit the availability of parallel execution because of using tree hashing: Kangaroo Twelve and Marsupilami Fourteen. [21]
The figure below shown one SHA-3 iteration.

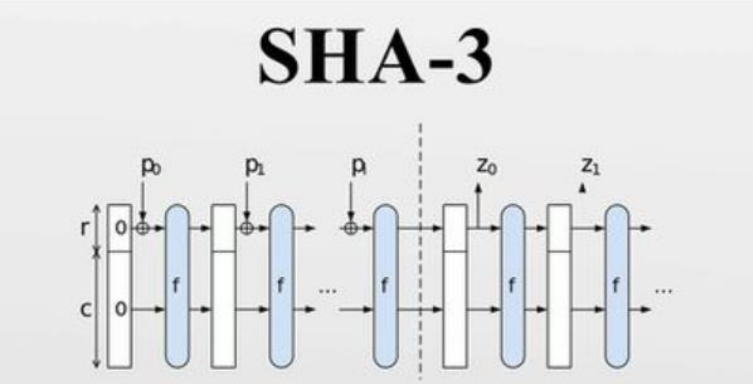


Figure 2.5-3:One SHA-3 Iteration [24]

2.5.2.1.1 How it works: [24]

The SHA-3 (Keccak) scheme consists of two stages:
Absorbing (absorption). The original message M is subjected to mnogoraundovaya(is a transliteration of the Russian word "многораундовая," which means "multi-round) the permutation f.

Squeezing (pressing). The output of the resulting permutations of the value z.
The function of Keccak consists of the following:

```
Keccak[r,c](M) {
Initialization and padding
for(int x=0; x<5; x++)
   for(int y=0; y<5; y++)
      S[x,y] = 0;  P = M || 0x01 || 0x00 || … || 0x00;
      P = P xor (0x00 || … || 0x00 || 0x80);  //Absorbing phase  for all block Pi in P
for(int x=0; x<5; x++)
for(int y=0; y<5; y++)
    S[x,y] = S[x,y] xor Pi[x+5*y];
    S = Keccak-f[r+c](S);  //Squeezing phase
    Z = empty string;
 do {  for(int x=0; x<5; x++)
 for(int y=0; y<5; y++)
    if((x+5y)<r/w)
       Z = Z || S[x,y];
       S = Keccak-f[r+c](S) }
while output is requested
 return Z; }
```

The Absorbing step calculates the hash value.
And at the stage of Squeezing the output of the results until the required hash length is reached.
There absorbing is function:
Keccak-f[b](A) {for all i in 0…nr-1 A = Round[b](A, RC[i]) return A }
Here b is the value of the selected function.
And the Round () function is a pseudo-random permutation applied on each round. The number of rounds "' nr '" is calculated from the values of r and c.
The operations performed on each round represent the following function:

```
Round[b](A,RC) {
 θ step
for(int x=0; x<5; x++)
C[x] = A[x,0] xor A[x,1] xor A[x,2] xor A[x,3] xor A[x,4];
for(int x=0; x<5; x++)    D[x] = C[x-1] xor rot(C[x+1],1);
    for(int x=0; x<5; x++)    A[x,y] = A[x,y] xor D[x];
      ρ and π steps  for(int x=0; x<5; x++)
for(int y=0; y<5; y++)
 B[y,2*x+3*y] = rot(A[x,y], r[x,y]);
 χ step  for(int x=0; x<5; x++)
for(int y=0; y<5; y++)      A[x,y] = B[x,y] xor ((not B[x+1,y]) and B[x+2,y]);
 ι step  A[0,0] = A[0,0] xor RC
 return A }
```

The SHA-3 digest size is always 512 bits, and thanks to hashing function guidelines, a minor change in the input string generates a drastically different digest. This is essential to prevent similar hash generation as much as possible, also known as a hash collision. The table below shows that no matter how much the size of the entered text changes, the output size remains constant

input text	**SHA-3 hash results**
Cryptography	01c888e1b6b32ee897bb0d9a2e30f8dfd3fdaff7d638f6f3bcf8ef500cf0bbe197b98aea768750aa0cb640e87a7b0544a031c58180ba6202865f9d71908210d9
COMPARISON BETWEEN HASHING ALGORITHMS	7b566ab10b174ae42aadf1960487fea548c5b20d79649a5e06fffbecf3ad14d63be633294add8b6dc3336837fde27dd2859a172cdeeb46f464ff2fbda7b0f855

Table 2.5-1: SHA-3 Hashing Examples

2.5.2.2 SHA-256:

SHA 256 is a part of the SHA 2 family of algorithms, where SHA stands for Secure Hash Algorithm. Published in 2001, it was a joint effort between the NSA and NIST to introduce a successor to the SHA 1 family, which was slowly losing strength against brute force attacks.

The significance of the 256 in the name stands for the final hash digest value, i.e. irrespective of the size of plaintext/cleartext, the hash value will always be 256 bits.[25]

2.5.2.2.1 SHA-256 Algorithm :23

the words on for size 32 bits in this algorithm. We have 64 words of turns of 32-b$W_0, W_1, \cdots, W_{63}$ ariables a, b, c, d, e, f, g, h of 32 bits and two auxiliary variables of 32 bits T_1 T_2. The program is then the following:

SHA-256:

Entrée : le message préparé et découpé ;

pour $(i = 1 ; i <= N ; i++)$

début

$W_t = M_t^{(i)}$ *pour* $(t = 0 ; t <= 15 ; t++)$;

$W_t = \sigma_1^{(256)}(W_{t-2}) + W_{t-7} + \sigma_0^{(256)}(W_{t-15}) + W_{t-16}$ *pour* $(t = 16 ; t <= 63 ; t++)$;

$a = H_0^{(i-1)}; b = H_1^{(i-1)}; c = H_2^{(i-1)}; d = H_3^{(i-1)}$;

$e = H_4^{(i-1)}; f = H_5^{(i-1)}; g = H_6^{(i-1)}; h = H_7^{(i-1)}$;

pour $(t = 0 ; t <= 63 ; t++)$

début

$T_1 = h + \Sigma_1^{(256)}(e) + Ch(e, f, g) + K_t^{(256)} + W_t$;

$T_2 = \Sigma_0^{(256)}(a) + Maj(a, b, c)$;

$h = g ; g = f ; f = e ; e = d + T_1$;

$d = c ; c = b ; b = a ; a = T_1 + T_2$;

fin

$H_0^{(i)} = a + H_0^{(i-1)}$; $H_1^{(i)} = b + H_1^{(i-1)}$;

$H_2^{(i)} = c + H_2^{(i-1)}$; $H_3^{(i)} = d + H_3^{(i-1)}$;

$H_4^{(i)} = e + H_4^{(i-1)}$; $H_5^{(i)} = f + H_5^{(i-1)}$;

$H_6^{(i)} = g + H_6^{(i-1)}$; $H_7^{(i)} = h + H_7^{(i-1)}$;

Fin

Sortie: $H_0^{(N)} || H_1^{(N)} || H_2^{(N)} || H_3^{(N)} || H_4^{(N)} || H_5^{(N)} || H_6^{(N)} || H_7^{(N)}$.

The SHA-256 digest size is always 256 bits, and thanks to hashing function guidelines, a minor change in the input string generates a drastically different digest. This is essential to prevent similar hash generation as much as possible, also known as a hash collision. The table below shows that no matter how much the size of the entered text changes, the output size remains constant.

input text	SHA-256 hash results
Cryptography	b584eec728548aced5a66c0267dd520a00871b5e7b735b2d8202f86719f61857
COMPARISON BETWEEN HASHING ALGORITHMS	eaffb07942466b28de2e693853559d00458a2334a15ecaf07206d592981a79ed

Table 2.5-2: SHA-256 Hashing Examples

2.5.2.3 MD5: [20]

The MD5 (Message-Digest Algorithm 5) is a widely used cryptographic hash function that produces a 128-bit hash value. It was designed by Ronald Rivest in 1991 to replace the earlier hash function MD4, and was specified in 1992 as RFC 1321.
The MD5 hash is typically represented as a sequence of 32 hexadecimal digits. Even a small change in the message will (with overwhelming probability) result in a mostly different hash, due to the avalanche effect.
MD5 is no longer considered secure for applications like digital signatures that rely on collision resistance. However, it is still used in various non-cryptographic applications like data integrity checks and partial password verification
It used to generate a 128-bit digest from a string of any length. It represents the digests as 32 digit hexadecimal numbers. the figure 2.5-4 show.

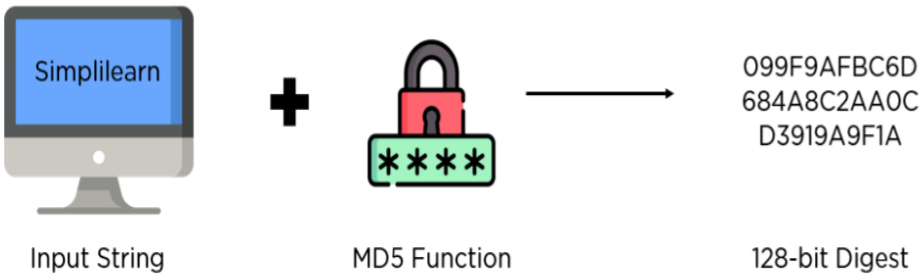


Figure 2.5-4:MD5 algorithm [20]

Ronald Rivest designed this algorithm in 1991 to provide the means for digital signature verification. Eventually, it was integrated into multiple other frameworks to bolster security indexes.

The table below shows that no matter how much the size of the entered text changes, the output size remains constant.

input text	md5 hash results
Cryptography	64ef07ce3e4b420c334227eecb3b3f4c
COMPARISON BETWEEN HASHING ALGORITHMS	4ec602277613706086c8a76becc5f80d

Table 2.5-3: MD5 Hashing Examples

The digest size is always 128 bits, and thanks to hashing function guidelines, a minor change in the input string generates a drastically different digest. This is essential to prevent similar hash generation as much as possible, also known as a hash collision.

2.5.2.3.1 How MD5 works: [20]

There are four major sections of the algorithm:

- Padding Bits:

When you receive the input string, you have to make sure the size is 64 bits short of a multiple of 512. When it comes to padding the bits, you must add one (1) first, followed by zeroes to round out the extra characters. As figure below show.

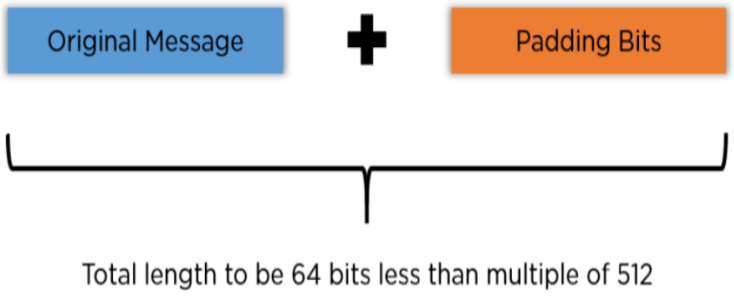


Figure 2.5-5:Padding bits [20]

- Padding Length:

You need to add a few more characters to make your final string a multiple of 512. To do so, take the length of the initial input and express it in the form of 64 bits. On combining the two, the final string is ready to be hashed. As figure below show.

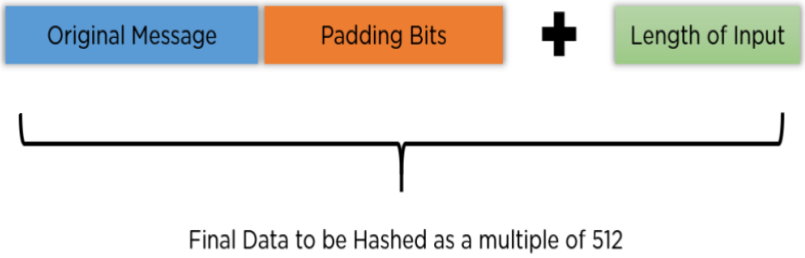


Figure 2.5-6:Input data to be hashed [20]

- Initialize MD Buffer:

The entire string is converted into multiple blocks of 512 bits each. You also need to initialize four different buffers, namely A, B, C, and D. These buffers are 32 bits each and are initialized as follows:

A = 01 23 45 67
B = 89 ab cd ef
C = fe dc ba 98
D = 76 54 32 10

- Process Each Block:

Each 512-bit block gets broken down further into 16 sub-blocks of 32 bits each. There are four rounds of operations, with each round utilizing all the sub-blocks, the buffers, and a constant array value.
This constant array can be denoted as T[1] -> T[64].
Each of the sub-blocks are denoted as M[0] -> M[15]. As figure below show.

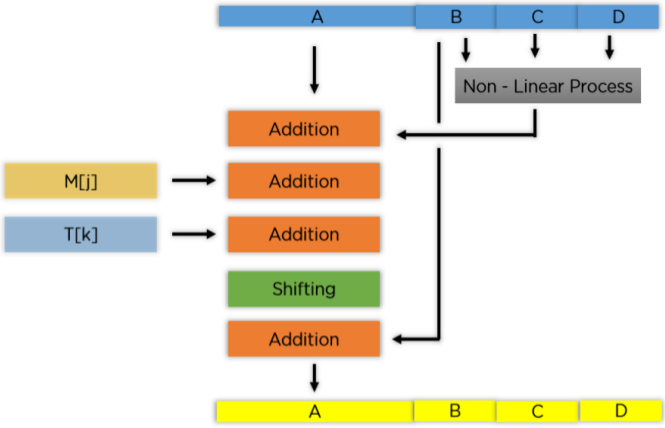


Figure 2.5-7:Data Flow with Intermediate States [20]

According to the image above, you see the values being run for a single buffer A. The correct order is as follows:

It passes B, C, and D onto a non-linear process.
The result is added with the value present at A.
It adds the sub-block value to the result above.
Then, it adds the constant value for that particular iteration.
There is a circular shift applied to the string.
As a final step, it adds the value of B to the string and is stored in buffer A.
The steps mentioned above are run for every buffer and every sub-block. When the last block's final buffer is complete, you will receive the MD5 digest.

The non-linear process above is different for each round of the sub-block.
Round 1: (b AND c) OR ((NOT b) AND (d))
Round 2: (b AND d) OR (c AND (NOT d))
Round 3: b XOR c XOR d
Round 4: c XOR (b OR (NOT d))
With this, you conclude the working of the MD5 algorithm.

2.6 Conclusion:

It has become clear that modern encryption serves as a cornerstone of secure communications and data protection in the modern digital world. By understanding the modern cryptographic principles and techniques presented here, we find that the different algorithms presented have varying strengths and weaknesses in terms of security, performance, and key management. This enables us to make informed

decisions about securing our sensitive information and protecting it from unauthorized access or modification.

This chapter has laid the foundation for understanding the basic tools of modern cryptography, in the next chapter, we will explore in detail the concept and architecture of our software system (website) in order to get a preview of Chapter 5 of Comparing cryptography Algorithms. We will delve deeper into these considerations and provide comparisons in choosing the optimal encryption method for security needs.

CHAPTER 3

3 Conception &Architecture of system

3.1 Introduction:

In this chapter, we attempted to create a cryptography algorithms comparison website for academic study that compares and evaluates different encryption methods to help you choose the most suitable one for your specific needs. So, we'll go through the overall architecture of our system, as well as the various diagrams. The application will then be realized. This chapter is divided into two main sections, which are as follows:

- The first part that talks about the architecture of our system, the comparison application itself as well as the various diagrams, such as the use case diagrams and the sequence diagram...etc.
- In Second part, we will begin with an overview of the tools used to achieve our system Development tools and some screenshots on the app Implemented.

and we will conclude with a conclusion

3.2 Conception:

3.2.1 Problem:

To achieve these objectives, this thesis addresses the following issues: What are the main strengths and weaknesses of cryptography algorithms? How does the performance of major cryptographic algorithms compare in terms of encryption/decryption time and resource consumption? Which cryptography algorithms offer the best performance for use in various scenarios?

3.2.2 Objectif:

The main objectives of this research are to provide accurate and detailed information to help make informed decisions regarding the choice of cryptography algorithms and to clarify the advantages and disadvantages of each algorithm in order to better understand their strengths and limitations empowering you to make informed decisions about securing your data.

3.2.3 Overall operation:

In order to design our application, we will represent it by a defined modeling language, we will be using the Unified Modeling Language (UML). UML is a standardized modeling language that empowers developers to define, visualize, build, and document the various components of a software system. This language ensures the scalability, security, and robustness of the system during execution. As a crucial aspect of object-oriented software development, UML employs graphic notation to create clear and concise visual models of software systems.

There are many types of UML diagrams, we choose from them use case diagram and Sequence diagram

3.2.3.1 Use case diagram

A use case diagram depicts the interactions between actors and use cases as well as the functional needs of the system. You can use them to:
• Describe the functional requirements of the system
• Describe how external things (actors) interact at the system boundary
• Describe the system response

Our use case diagram shown in the figure below.:

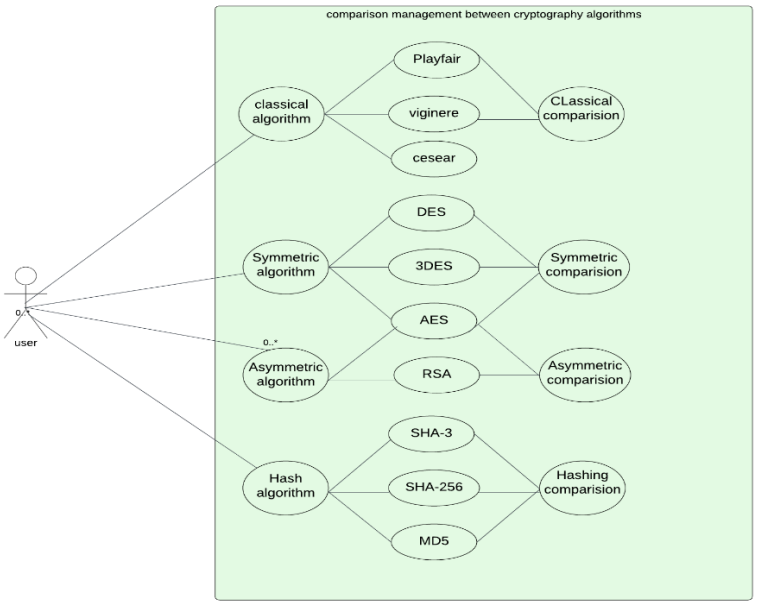


Figure 3.2-1:Use Case Diagram of comparison between cryptography algorithms

3.2.3.2 Sequence Diagram:

A sequence diagram is an organized representation of behavior as a series of sequential steps over time. You can use it for:
• Describe workflow, message passing, and how elements generally collaborate over time to achieve a result.
• Capture the flow of information and responsibility throughout the system, early in the analysis; Messages between items eventually become method calls on the Class form.
• Making illustrative models for use case scenarios. By creating a sequence diagram with an actor and items included in a Use Case, you can model the sequence of steps that the user and system take to complete the required tasks

A sequence diagram is used in this chapter to depict how items interact with one another over time.
The figure below shown our sequence diagram.

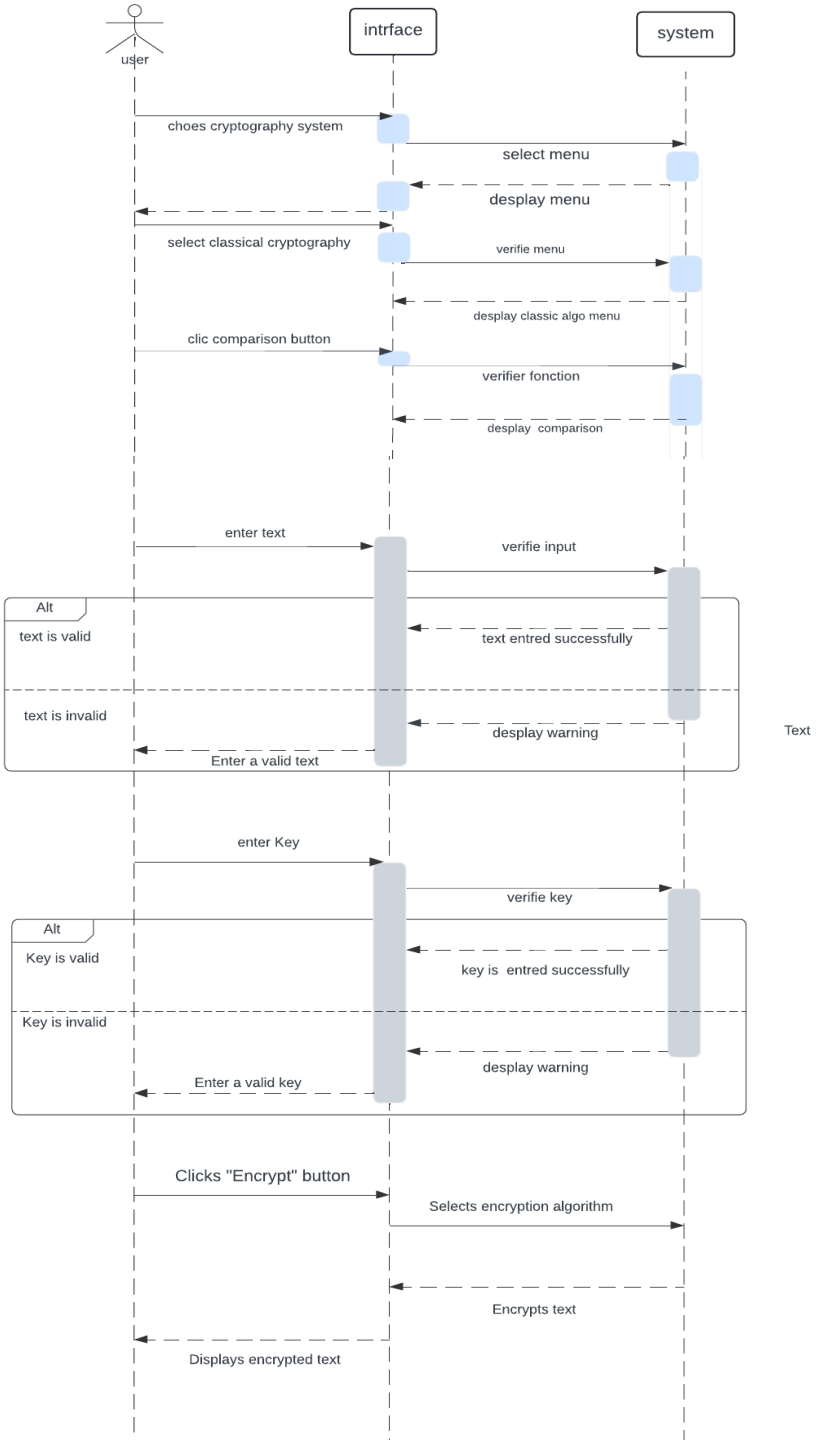


Figure 3.2-2:sequence Diagram of classical comparison

In this part, we've outlined our system's design using UML sequence and use case diagrams. Now that the foundation is laid, the next part will delve into the practical implementation and realization of this system.

3.3 Architecture of system:

3.3.1 Framework:

Bootstrap:

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites. Bootstrap is a free front-end framework for faster and easier web development. It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others, as well as optional JavaScript plugins Bootstrap also gives you the ability to easily create responsive designs[29]

3.3.2 Software environment:

Visual Studio Code:

Visual Studio Code (VS Code) is a free, open-source code editor developed by Microsoft. It is designed to be lightweight and powerful, supporting a wide range of programming languages and extensions [28].VS Code helps you be instantly productive with syntax highlighting, bracket matching, auto-indentation, box selection, code snippets, and more.

Modelio:

Modelio is an open-source modeling environment developed by Modeliosoft. It supports UML2, BPMN and ArchiMate standards. Modelio offers a wide range of standards-based features for software developers, analysts, designers, business architects and system architects.

3.3.3 Language:

HTML: (Hypertext Markup Language):

HTML (HyperText Markup Language) is the standard markup language used to structure and format content on the web. It is used to define the structure and content of web pages, including headings, paragraphs, images, links, and other elements[30].Without HTML, a browser would not be able to display text as elements or load images or other elements.

CSS:

CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation of a document written in a markup language like HTML[31]. It is used to control the layout, color, font, and other visual aspects of a web page

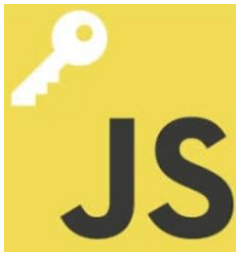
JavaScript:

JavaScript, often abbreviated as JS, is a fundamental programming language and a core technology of the Web. JavaScript is a lightweight interpreted or just-in-time compiled language with first-class functions, supporting multiple paradigms like object-oriented, imperative, and declarative styles. It is prototype-based, single-threaded, and dynamic, offering dynamic capabilities such as runtime object construction, variable parameter lists, and dynamic script creation.[27]

3.3.4 Another tool:

jQuery:

jQuery is a popular JavaScript library that simplifies HTML DOM manipulation, event handling, and AJAX. It can concise and cross-browser compatible API for common web development tasks, making it a popular choice among web developers.



CryptoJS:

CryptoJS is A popular JavaScript library that provides a collection of standard and secure cryptographic algorithms implemented in JavaScript. It provides implementations of various hashing algorithms like MD5, SHA-1, SHA-2, and SHA-3, as well as symmetric encryption algorithms like AES, Triple DES, RC4, Rabbit, and PBKDF2. CryptoJS is designed to be fast and have a consistent and simple interface. It is commonly used in web applications for tasks like password hashing, cookie encryption, and protecting sensitive data.[32]

Word:

Word is one of the most widely used word processing software in the world. Belonging to Microsoft's office suite, Word allows you to write and format text documents.

3.3.5 Hardware environment:

When creating this website, we used the following equipment shown in this table:

	The characteristics
Processor	11th Gen Intel(R) Core (TM) i5-1155G7 @ 2.50GHz 2.50 GHz
RAM Memoirs	16 Go
Disque dur	512 Go SSD
GPU	Intel(R)Iris(R)Xe Graphics
Operating system	Windows 11

Table 3.3-1:The characteristics of the material used

3.3.6 Process of system:

3.3.6.1 Homepage:

The main page of our website is seen in Figure 3.3-1 The user doesn't need to register for an account in order to view this home page, It appears directly to the user. This homepage features three portals. The home page itself, our different cryptographic systems, and information about the website. This page is divided into two parts: An illustration that gives a visual overview of the website's content that focuses on cryptographic systems, and a paragraph gives a concise understanding of what the website is and the services it offers.

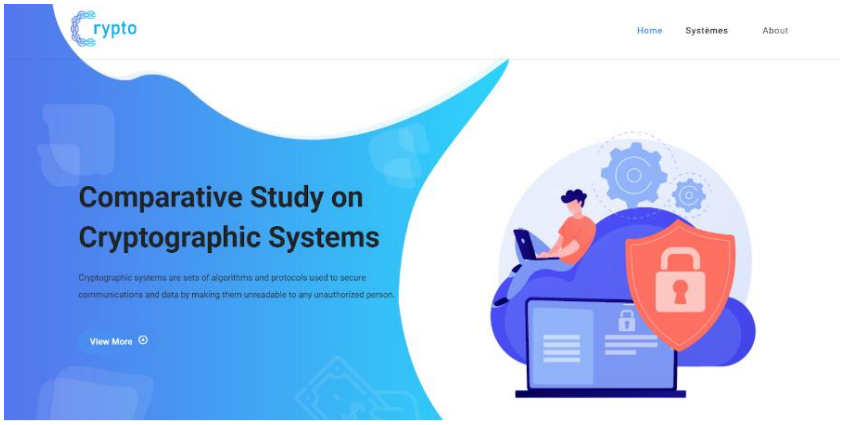


Figure 3.3-1:Homepage of the proposed system

3.3.6.2 Cryptography system category page:

This interface contains four main categories of cryptographic systems which is: Classic cryptography, symmetric cryptography, asymmetric cryptography and hashing algorithms, as seen in Figure 3.3-2 Each has a link to its own cryptographic algorithm used

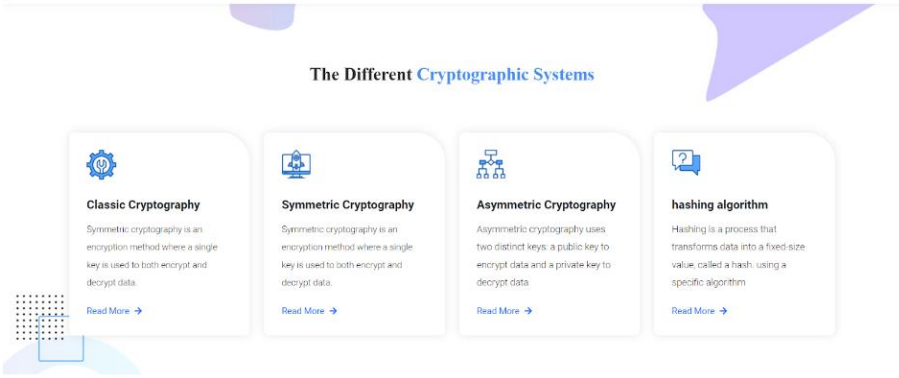


Figure 3.3-2:Cryptography system category page

3.3.6.3 Symmetric Cryptography:

The symmetric Cryptography page is shown in the figure 3.3-3. This page contains three different interfaces, each containing the title of the algorithm used with a small descriptive paragraph about the algorithm (DES, 3DES, AES).

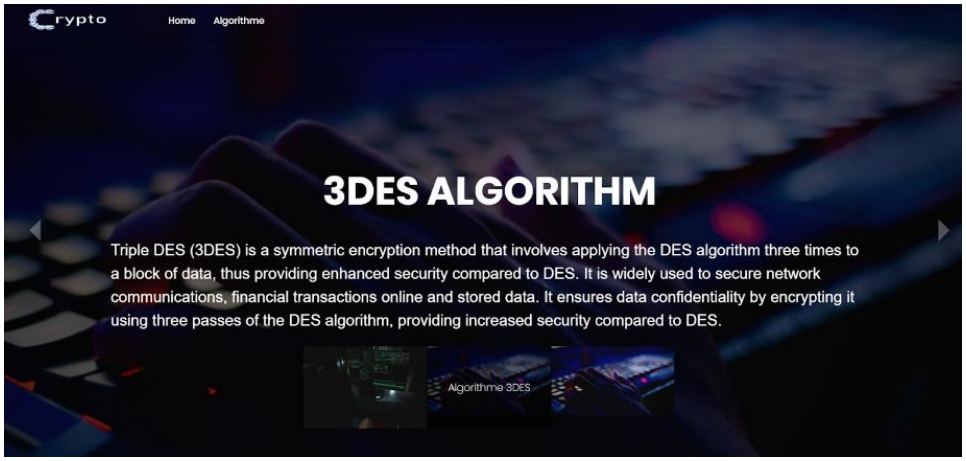


Figure 3.3-3:Symmetric Cryptography,3DES algorithm

3.3.6.4 Algorithm category:

This page gives the user to choose one of the cryptography algorithms by giving a small definition of the algorithm, this page enables him to enter his own data to encrypt or decrypt it for the comparison. we have an input for text and key, then an encryption or decryption button, after that we display the results (text encryption/decryption, execution time, complexity, memory space used, power consumption, performance). This is what the following figure show.

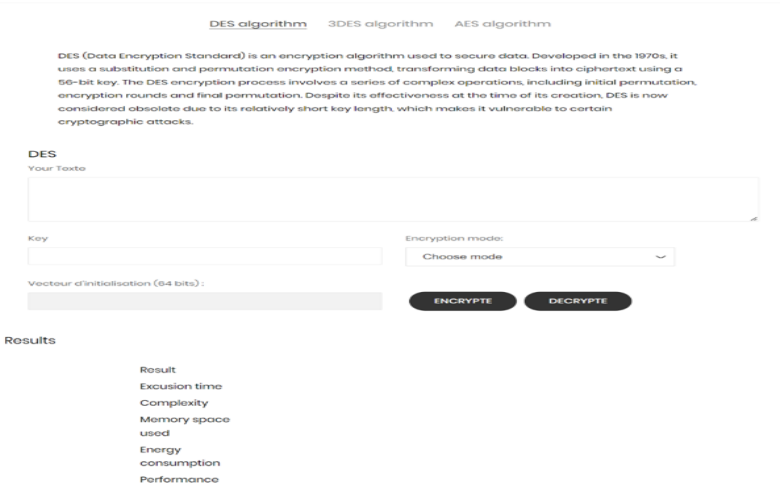


Figure 3.3-4:Symmetric algo DES

Algorithm comparison page:

The comparison page is shown in the figure 3.3-5. On this page, the user can choose and compare between algorithms, in order to know the result that each algorithm will give. It also enables him to display the comparison results in a table.in the figure below the comparison is between symmetric cryptography algorithm AES, DES and 3DES

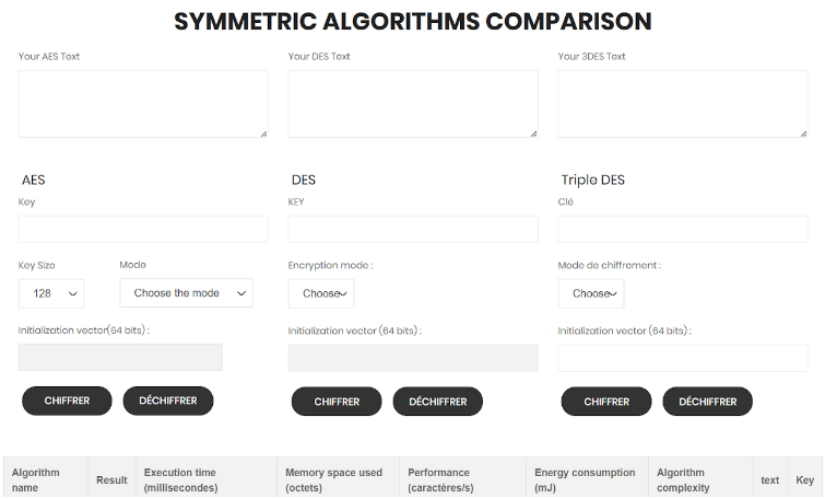


Figure 3.3-5:symmetric algorithms comparison

3.4 Conclusion:

In this chapter, we detailed the creation of a cryptography algorithms comparison website designed for academic purposes. The system's architecture was meticulously outlined, including the various diagrams essential for understanding the overall design and functionality of the application. We provided an in-depth look at use case diagrams and sequence diagrams, which are crucial for illustrating the interactions within the system.

This chapter has provided a thorough exploration of the design and implementation of a cryptography algorithms comparison website. By understanding the architectural framework and the tools employed. In the next chapter, we'll delve into cryptography Algorithm Comparison, and provide the comparisons' results, which will enable us to make informed decisions about securing data.

CHAPTER 4

4 Experimentation and evaluation of results

4.1 Introduction:

In the realm of cryptography and data security, the efficacy and performance of various algorithms are of paramount importance. This chapter delves into the experimental analysis and evaluation of several cryptographic algorithms, providing a comprehensive comparison across different categories. By systematically assessing the strengths and weaknesses of these algorithms, this study aims to offer valuable insights that can guide the selection and implementation of the most suitable cryptographic techniques for specific applications.

Comparison in this context, refers to the systematic process of evaluating two or more algorithms based on specific metrics such as performance, execution time, complexity, memory space used and Energy consumption. The goal is to identify the relative strengths and weaknesses of each algorithm, enabling an informed decision on which algorithm best meets the requirements of a given application.

4.2 Cryptography algorithm comparison:

We will analyze a range of cryptographic algorithms, including classic algorithms "Cesar Playfair and Vigenère", symmetric algorithms " AES, DES,3DES", asymmetric algorithms "RSA", and hashing algorithms "SHA-256, MD5, SHA-3. The choice of the most appropriate algorithm can significantly impact the security, efficiency, and overall performance of cryptographic systems in various applications, from secure communications to data integrity verification.

The results of encryption or decryption for each algorithm for the comparison is display on tables with the measure of each criterion.so to study the comparison between cryptography algorithm we just need to compare the results of the tables.

It is important to note that the results presented in these tables are for illustrative purposes only. Actual algorithm performance may vary depending on the hardware and software configuration used.

In this chapter, we performed several encryption and decryption operations for different algorithms. To prove the reliability of the results we obtained, we tested them by comparing them with the results obtained from many famous websites for encryption and decryption, using the same text and same key. which we will mention now:

Classical algorithms:

- Cesear: Caesar cipher [10]
- Vigenère: Chiffre de Vigenère [11]
- Playfair: Chiffre de Playfair [26]

Symmetric algorithms:

- AES: AES encryption / decryption [33]
- DES: DES encryption/decryption [34]
- 3DES: Triple DES Encryption and Decryption Online Tool. [35]

Asymmetric algorithms:

- RSA: Online RSA Key Generator [36]

Hashing algorithms:

- MD5: Online MD5 Hash Calculator [37]
- SHA-3: Online SHA3-256 Hash Calculator. [38]
- SHA-256: SHA-256 hash calculator [39]

4.3 Comparison Criteria:

Algorithms are evaluated based on five criteria:

- Execution time: Measures the time required to complete encryption or decryption processes.in other word, the time required to execute the algorithm, expressed in seconds.
- Memory space: The amount of memory required to execute the algorithm, expressed in bytes.

memoryUsed = performance.memory.usedJSHeapSize; // Get used memory space

- Performance: The throughput of the algorithm, expressed in characters/ second.

textLength / executionTime) * 1000).toFixed(6)

- Energy consumption: The amount of power consumed by the algorithm, expressed in millijoules.

executionTime * (memoryUsed / 1000)).toFixed(6)

ToFixed(6): Ensures that there are six digits after the decimal point. If the number has fewer than six digits after the decimal, zeros are added to make up the difference.

- Complexity: Assesses the algorithm's computational complexity and resource requirements.

4.4 Comparison between classical algorithms:

4.4.1 Comparison between Playfair & Vigenère using medium text:

TEXT 2: COMPARISON BETWEEN CLASSIC ALGORITHMS

Key 1: sic Result:

Playfair Encryption:	Vigenere *Encryption*:
bmkrctcikoihqyfwglimbiicabnemuaqfobv	UWOHITAAQF JGLEGWV EDIUKQE STIGZKLPOK
Playfair Decryption:	Vigenere Decryption:
comparisonbetwexenclassicalgorithmsz	COMPARISON BETWEEN CLASSIC ALGORITHMS

Table below shows the result of encryption and decryption of text 2 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Execution time (ms)	Memory space used (Mo)	Energy consumption (mj)	Performance (caractères/ms)	Algorithm complexity
Playfair(Encryption)	2.239	9.54	22392.82	15.183434920068308	O(n)
Vigenère (Encryption)	0.70	9.536743	178001999.999285	54.297779	O(114)
Playfair(Decryption)	2.508	9.54	25078.31	7.177518287094867	O(n)
Vigenère (Decryption)	0.47	9.536743	357015999.999046	81.232706	O(114)

Table 4.4-1: Comparison between Playfair & Vigenère using short Key & medium text

The execution time result is representing in the following figure.

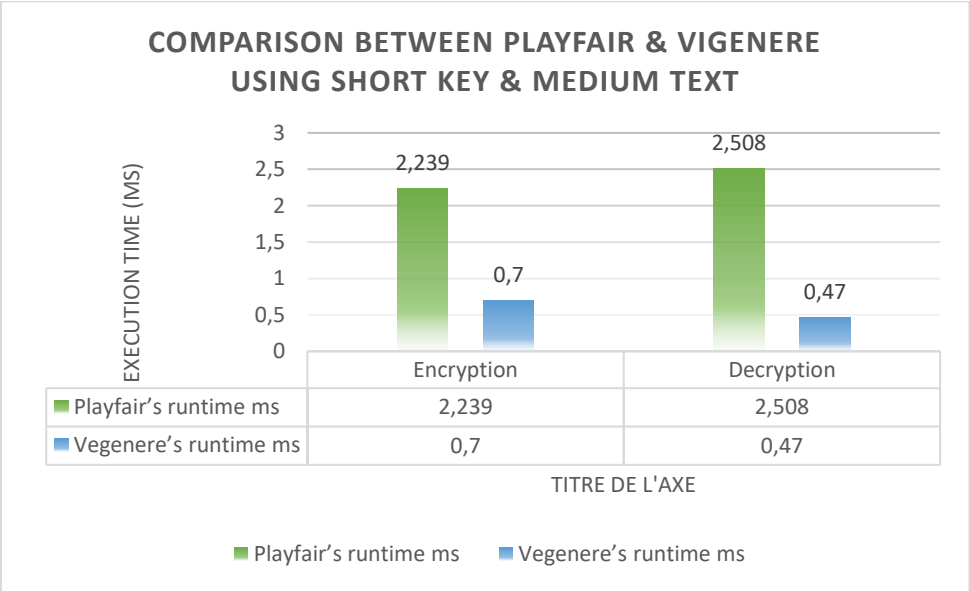


Figure 4.4-1:Histogram explain comparison between Playfair & Vigenère using short Key & Medium text

4.4.1.1 Observation:

Execution time: Playfair is slower than Vigenère for encryption and decryption

Memory space used: Playfair and Vigenère use approximately the same amount of memory.
Energy consumption: Vigenère consumes more energy than Playfair for encryption and decryption.
Performance: Vigenère is even more efficient than Playfair in terms of encryption and decryption.
Complexity of the algorithm: Playfair is more complex than Vigenère.

4.4.1.2 Conclusion:

Based on the evaluated metrics, Vigenère appears to be a better performing encryption algorithm than Playfair. It is faster, more energy efficient and offers better performance.

4.4.2 Comparison between Playfair & Vigenère using very long text

TEXT 4 : the size of the text used is : 4996 characters
Key 1 : sic
Result:
Playfair encryption: the result of the size of the text encrypted is : 4999 characters
Vigenère encryption: : the result of the size of the text encrypted is : 4999 characters

Table below shows the result of encryption and decryption of text 4 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution time (ms)	Memory space used (Mo)	Energy consumption (mj)	Performance (caractères/ms)	Algorithm complexity	Text
Playfair(Encryption)	idfxgqebapmuaydfapmu...	409.280	9.54	4092800.00	12.5	O(n)	Secret History: The Stor...
Vigenère (Encryption)	KMEJMV ZQULWTQ: B...	39.55	9.536743	168118000.000715	165.000000	O(19575)	SECRET HISTORY: TH...
Playfair(Decryption)	secrethistorythestoryofc...	360.000	9.54	3600000.00	7.25	O(n)	idfxgqebapmuaydfapmu...
Vigenère (Decryption)	SECRET HISTORY: TH...	38.38	9.536743	350595000.000000	170.000000	O(19575)	KMEJMV ZQULWTQ: B...

Table 4.4-2: Comparison between Playfair & Vigenere using short Key and very long text

The execution time result is representing in the following figure.

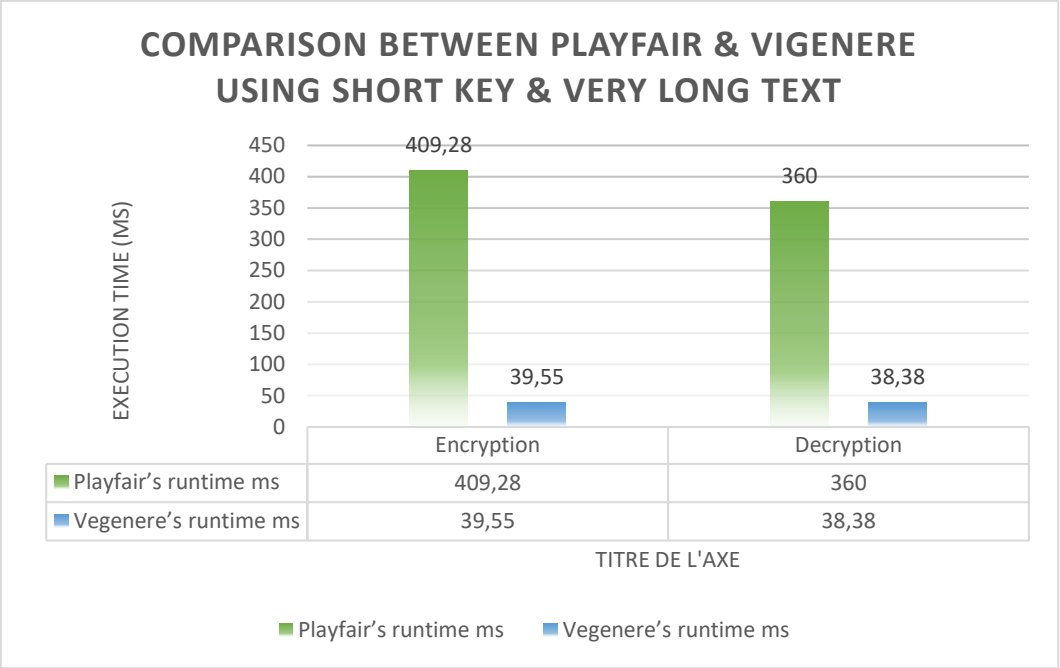


Figure 4.4-2:Histogram explain comparison between Playfair & Vigenere using short Key & very long text

4.4.2.1 Observation:

Execution time: Playfair is slower than Vigenère for encryption and decryption. The difference in Playfair execution time is greater for encryption than for decryption.
Memory space used: Playfair and Vigenère use approximately the same amount of memory.
Energy consumption: Vigenère consumes more power than Playfair for encryption and decryption.
Performance: Vigenère performs better than Playfair for encryption and decryption.
Algorithm complexity: Playfair is more complex than Vigenère.

Based on the evaluated metrics, Vigenère appears to be a better performing encryption algorithm than Playfair. It is faster, more energy consumption and offers better performance.

From the tables analyzed previously, we can observe that:
Playfair:
• Encryption execution time Longer than decryption execution time.
• Impact of text length: Execution time increases linearly with text length.

4.4.2.2 Explanation of the differences:

Playfair: Playfair's encryption operations are more complex than decryption operations, which explains the difference in execution time in case the txt is large

4.4.2.3 Conclusion:

The encryption execution time is always longer than the decryption execution time for both algorithms.
The difference in Playfair execution time between encryption and decryption, encryption is greater than decryption.

The choice between Playfair and Vigenère depends on safety and performance requirements. If speed is the priority, Vigenère is the obvious choice. However, if security is paramount and performance is not a crucial factor, Playfair may be an acceptable option.
Run times shown in the tables are examples and may vary depending on the implementation and hardware used.
Other factors, such as key length and character distribution in the text, can also affect performance.

- **Now we are going to change the key size:**

The results are recorded in the table below: we use the same text used before but with a long key “master two computer security and cryptography “

Short = Text 1, Medium = Text 2, Long = Text 3, Very Long = Text 4

Text Length	Playfair's Runtime (ms)	Vigenere's Runtime (ms)
Short	Encryption: 1.023, Decryption: 1.456	Encryption : 0.16, Decryption: 0.14
Medium	Encryption:2.283 , Decryption:2.411	Encryption: 0.57, Decryption: 0.41
Long	Encryption: 7.561, Decryption: 7.308	Encryption: 2.04, Decryption: 1.76
Very Long	Encryption: 408.880, Decryption : 359.586	Encryption: 39.47, Decryption: 38.31

Table 4.4-3: Execution time of encryption and decryption of Playfair and Vigenère for different text sizes

We obtained almost the same results as before, with a slight difference in execution time due to the size of the key.

4.5 Comparison between symmetric algorithms:

In the code which we use to carry out the encryption and decryption of the algorithm, it specifies the key if it’s less than the size (AES {128,192,256}, DES {64}, 3DES {192}) it continues with 0s.
Example: AES: SICM2INF => SICM2INF00000000.

Remark:

- if we use the same encryption key of DES, we will obtain the same result, and this is because 3DES requires key 192 if the key is less than 64 bytes it gives the same result like DES for the encryption and decryption not for the other factors.

Example:
For AES: SICM2INF => SICM2INF00000000
For 3DES: SICM2INF => SICM2INF0000000000000000

4.5.1 Comparison between symmetric algorithms AES<128:

The following results are used in the test, AES key less than the specified size to see the difference. but DES and 3DES here is taking the real key size.

Here the comparison (AES key size < 128) is just for ECB mode.

Text 1: COMPARISON
Key 1: the key is represented in the following table:

Key size	< 128	=64	=192
AES	SICM2INF		
DES		SICM2INF	
3DES			SICM2INFORMATIQUECLASS12

Table 4.5-1:Table of keys chosen from symmetric algos

Result:

> AES Encryption : KuAlQFI6pY4fEkCSvQR83w==

> DES Encryption : l9bk3kyK0mB67u2Qa7aoBQ==

> 3DES Encryption : xk5E4mcaEcxPksVT+Mgp7Q==

Table below shows the result of encryption and decryption of text 1 with the selected key, and the comparison results of the algorithms based on the previous metrics

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
AES Encryption	KuAIQFi6pY4/EkCSvQ...	1.00	10000000	1000.00	10000.00	O(n^2)	COMPARISON	SICM2INF
DES Encryption	19bk3kyK0mB67u2Qa7...	1.60	1250000.00	6249.99	16000.00	O(n^2)	COMPARISON	SICM2INF
3DES Encryption	xk5EAmcaEcxPksVT+.	1.30	1250000	7692.30	12999.99	O(n^2)	COMPARISON	SICM2INFORMATIQUE
AES Decryption	COMPARISON	0.50	10000000	50000.00	5000.00	O(n^2)	KuAIQFi6pY4/EkCSvQ...	SICM2INF
DES Decryption	COMPARISON	1.30	1250000.00	19230.76	13000.00	O(n^2)	19Bk3kyK0mB67u2Qa7...	SICM2INF
3DES Decryption	COMPARISON	0.90	1250000	27777.77	8999.99	O(n^2)	xk5EAmcaEcxPksVT+.	SICM2INFORMATIQUE

Figure 4.5-1:Comparison between symmetric algorithms (AES, DES,3DES) short text.

The execution time result is representing in the following figure.

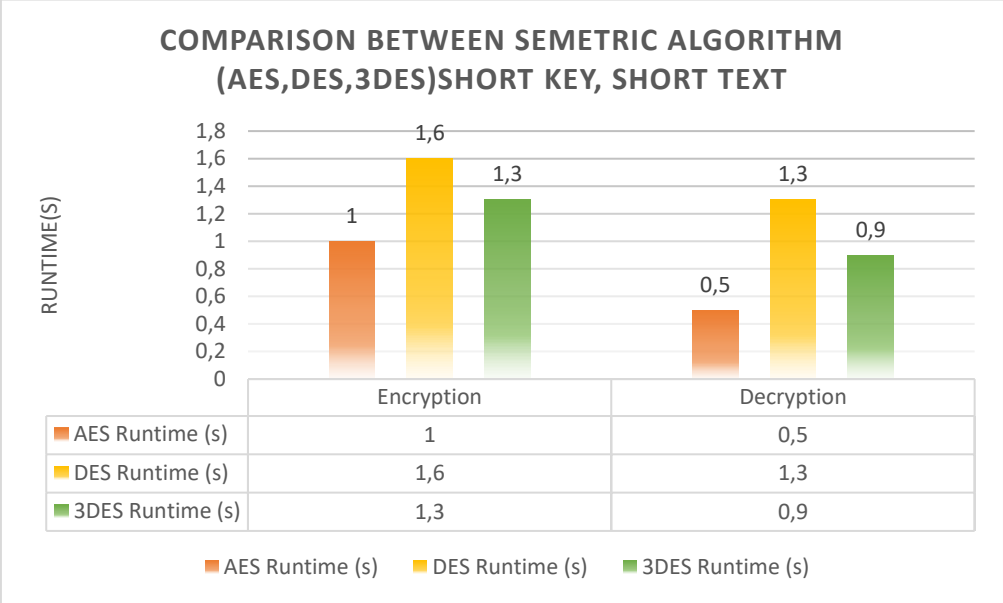


Figure 4.5-2:histogram explain comparison between AES, DES, 3DES using short Key and short text

4.5.1.1 Observations:

The table compares the performance of three symmetric encryption algorithms: AES, DES, and 3DES.
AES is the fastest algorithm for both encryption and decryption.
DES is the slowest algorithm for both encryption and decryption.
3DES is slower than AES but faster than DES for both encryption and decryption.
AES uses the least amount of memory for both encryption and decryption.
DES uses the most amount of memory for both encryption and decryption.
3DES uses more memory than AES but less than DES for both encryption and decryption.

4.5.1.2 Conclusion:

Based on the observations above, I would recommend using AES for most applications. AES is the most efficient algorithm in terms of both performance and memory consumption. However, if there are concerns about the security of AES, then 3DES could be used as a compromise between security and performance. DES should only be used if there are very strict performance requirements and security is not a major concern.

Text 3: Symmetric encryption is an encryption method where the same key is used to both encrypt and decrypt a message. Unlike asymmetric encryption, which uses a pair of keys (public and private), symmetric encryption provides faster processing speed because it only requires a single key for encryption and decryption operations.

Key 1: same key
Result:

> AES Encryption :
> /Zfr8TsSQk39TQ1I0634J4uSp99YmUKqva1LAeJY/q6w4GcPblKiZvWSlyUe2liK

> DES Encryption :
> 0Gw4cJD2U7x6zJFZuTDdFLLrbbeOs42WAGamYhsGiIlChcyKTyCaPg==

> 3DES Encryption :
> 3ncjbifTOmBZYS8hk4G8Y9vplHBiAx4vxRLJbOEVbnMHRR6ATTHVUA==

Table below shows the result of encryption and decryption of text 3 with the selected key, and the comparison results of the algorithms based on the previous metrics

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
AES Encryption	Vin2V07B2yR1P/LE20...	2.00	10000000	151000.00	20000.00	O(n^2)	Symmetric encryptions.	SICM2INF
DES Encryption	3lyrtRu6NJ7bfFWukW...	5.50	1250000.00	585445.45	55000.00	O(n^2)	Symmetric encryptions.	SICM2INF
3DES Encryption	UaURKSnG1thWIbBW...	4.00	1250000	80500.00	40000.99	O(n^2)	Symmetric encryptions.	SICM2INFORMATIQUE
AES Decryption	Symmetric encryptions.	1.00	10000000	449000.00	10000.00	O(n^2)	Vin2V07B2yR1P/LE20.	SICM2INF
DES Decryption	Symmetric encryptions...	2.60	1250000.00	159615.38	26000.00	O(n^2)	3lyrtRu6NJ7bfFWukW...	SICM2INF
3DES Decryption	Symmetric encryptions.	2.90	1250000	152063.96	28999.99	O(n^2)	UaURKSnG1thWIbBW...	SICM2INFORMATIQUE

Table 4.5-2:Comparison between symmetric algorithms (AES, DES,3DES) using long text

we obtained the same result from the previous table despite we change the text size

4.5.1.3 General conclusion:

Significant Speed Advantage for AES: The tables likely shows a clear advantage for AES in terms of execution time for both encryption and decryption in all text sizes to DES and 3DES. AES is a more modern algorithm designed for efficiency, resulting in faster processing of the all the text data.

3DES - The Middle Ground: While not as fast as AES, 3DES likely exhibits a noticeable improvement in speed compared to DES. This is because DES applies the encryption algorithm only once, whereas 3DES applies it three times, offering some additional security but at the cost of some speed.

Minimal Performance Impact: all the text size used in the tables likely minimizes the overall performance differences between the algorithms. For larger texts, the execution time gap between AES and the other two algorithms would become more pronounced.

4.5.2 Comparison between symmetric algorithms Mode ECB:

For each algorithm we use the real key size (for AES 128, DES 64, 3DES 192)

Key : the key is represented in the following table:

Key size	128	=64	=192
AES	SIC INFORMATIQUE		
DES		SICURITE	
3DES			Triple DES Algorithme M2

Table 4.5-3: keys chosen from symmetric algos mode ECB

Txt1 : COMPARISON .
Result:
AES: WhbRh+Ih4sw6WarkbS/Jew==
DES: SLtlx9ObvO9Tm7tMHkn2WQ==
3DES: jPu35wF01qRVYc00EjdNuQ==

Table below shows the result of encryption and decryption of text 1 with the selected key, and the comparison results of the algorithms based on the previous metrics

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
AES Encryption	WhbRh+|h4sw6WarkbS...	2.60	1250000.00	3846.15	25999.99	O(n^2)	COMPARISON	SICURITE
AES Decryption	COMPARISON	1.60	1250000.00	15625.00	19999.99	O(n^2)	WhbRh+|h4sw6WarkbS...	SICURITE
DES Encryption	SLtx90bv09Tm7MHk...	0.70	10000000	14285.71	6999.99	O(n^2)	COMPARISON	SIC INFORMATIQUE
DES Decryption	COMPARISON	0.50	10000000	50000.00	5000.00	O(n^2)	Vin2V07B2yR1P/LE20.	SIC INFORMATIQUE
3DES Encryption	jPu35wF01qRVYc00Ejd.....	1.60	1250000.00	6250.00	6999.99	O(n^2)	COMPARISON	Triple DES Algorithme M2
3DES Decryption	COMPARISON	0.80	1250000.00	31249.99	8000.00	O(n^2)	UaURKSnG1thWIbBW...	Triple DES Algorithme M2

Table 4.5-4: Comparison between AES, DES,3DES in « ECB mode », using short text

The execution time result is representing in the following figure.

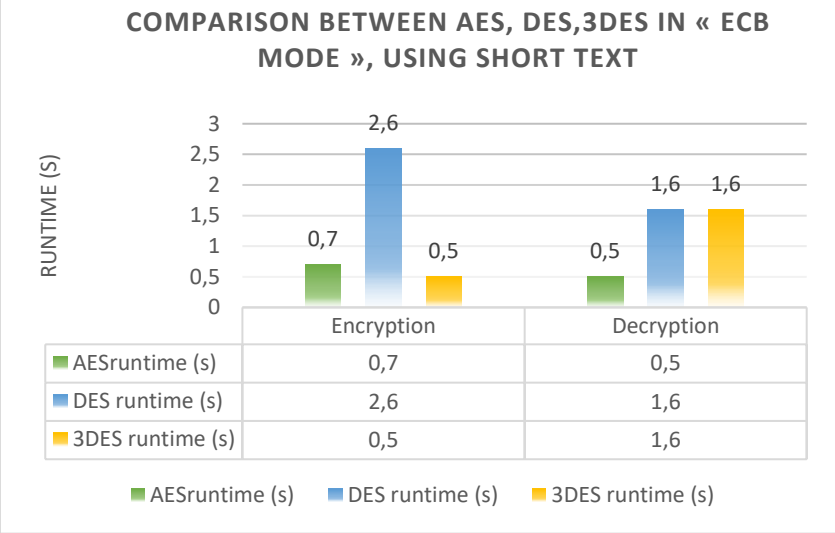


Figure 4.5-3:Comparison between AES, DES,3DES in « ECB mode », using short text

Txt3
Result:
AES: the size of the encrypted text is: 448 characters
DES: the size of the encrypted text is: 440 characters
3DES: the size of the encrypted text is: 440 characters

Table below shows the result of encryption and decryption of text 3 with the selected key, and the comparison results of the algorithms based on the previous metrics

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
3DES Encryption	5QsqFJOJ7K1yyGAK}j...	5.00	1250000.00	64400.00	50000.00	O(n^2)	SYMMETRIC ALGORIT...	Triple DES Algorithme M2
3DES Decryption	Symmetric encryption is...	4.30	1250000.00	102558.13	43000.00	O(n^2)	5QsqFJOJ7K1yyGAK}j...	Triple DES Algorithme M2
DES Encryption	WQ1haaOG1x/zOzbqQ...	3.60	1250000.00	89444.44	35999.99	O(n^2)	SYMMETRIC ALGORIT...	SICURITE
DES Decryption	Symmetric encryption is...	3.00	1250000.00	147000.00	30000.00	O(n^2)	WQ1haaOG1x/zOzbqQ...	SICURITE
AES Encryption	PMhR+aeSRzVISxGZQ...	1.60	10000000	201250.00	15999.99	O(n^2)	SYMMETRIC ALGORIT...	SIC INFORMATIQUE
AES Decryption	Symmetric encryption is...	1.20	10000000	374166.66	11999.99	O(n^2)	PMhR+aeSRzVISxGZQ...	SIC INFORMATIQUE

Table 4.5-5:Comparison between AES, DES,3DES in « ECB mode », using long text

The execution time result is representing in the following figure.

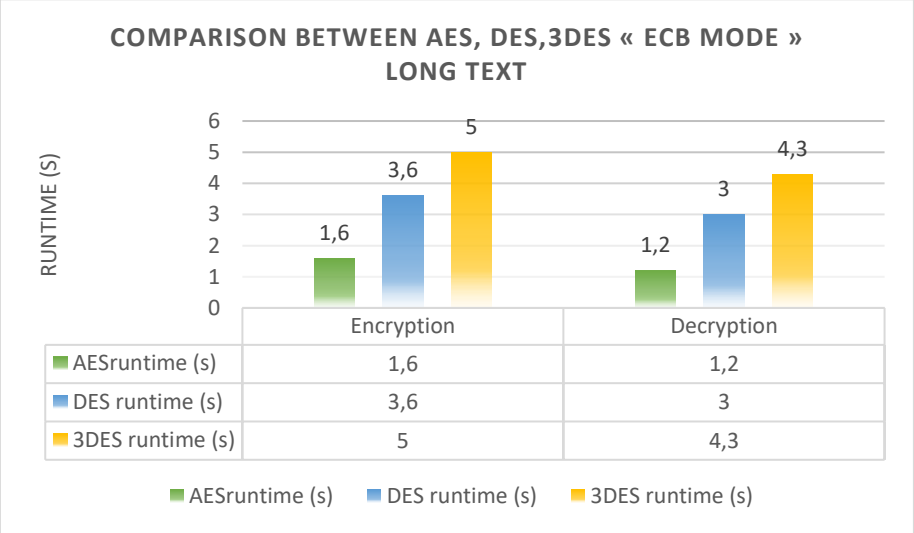


Figure 4.5-4:histogram explain comparison between AES, DES, 3DES using Long text in ECB mode

4.5.2.1 Conclusion:

In these tables we have almost the same results except the we see a small increase in the execution time that due to the key used for these ones.
The execution time of symmetric decryption algorithms is greater than that of encryption of very long texts.

4.5.3 Comparison between symmetric algorithms mode CBC:

For the CBC mode we use an IV (initiation vector). Its value is fixed for all the test.
IV = CRYPTO24
Key: the key is represented in the following table:

Key size	=128	=64	=192
AES	SIC INFORMATIQUE		
DES		SICURITE	
3DES			Triple DES Algorithme M2

Table 4.5-6: keys chosen from symmetric algos in CBC mode

Table below shows the result of encryption and decryption of text 2 with the selected key, and the comparison results of the algorithms based on the previous metrics

Text 2:

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
AES Encryption	IjC+nQcSze5AOI7KP9...	2.30	10000000	13913.04	23000.00	O(n^2)	SYMMETRIC ALGORIT...	SIC INFORMATIQUE
AES Decryption	SYMMETRIC ALGORIT...	1.60	10000000	40000.00	15999.99	O(n^2)	IjC+nQcSze5AOI7KP9...	SIC INFORMATIQUE
DES Encryption	1QE9s Tro/4CavB3MUFb...	3.40	1250000.00	9411.76	33999.99	O(n^2)	SYMMETRIC ALGORIT...	SICURITE
DES Decryption	SYMMETRIC ALGORIT...	1.80	1250000.00	311111.11	17999.99	O(n^2)	1QE9s Tro/4CavB3MUFb.	SICURITE
3DES Encryption	pNiLe+4|Vrd|SDvqOIVT...	1.50	1250000.00	21333.33	15000.00	O(n^2)	SYMMETRIC ALGORIT...	Triple DES Algorithme M2
3DES Decryption	SYMMETRIC ALGORIT...	1.00	1250000.00	56000.00	10000.00	O(n^2)	pNiLe+4|Vrd|SDvqOIVT...	Triple DES Algorithme M2

Table 4.5-7:Comparison between AES, DES,3DES in « CBC mode », using medium text

The execution time result is representing in the following figure.

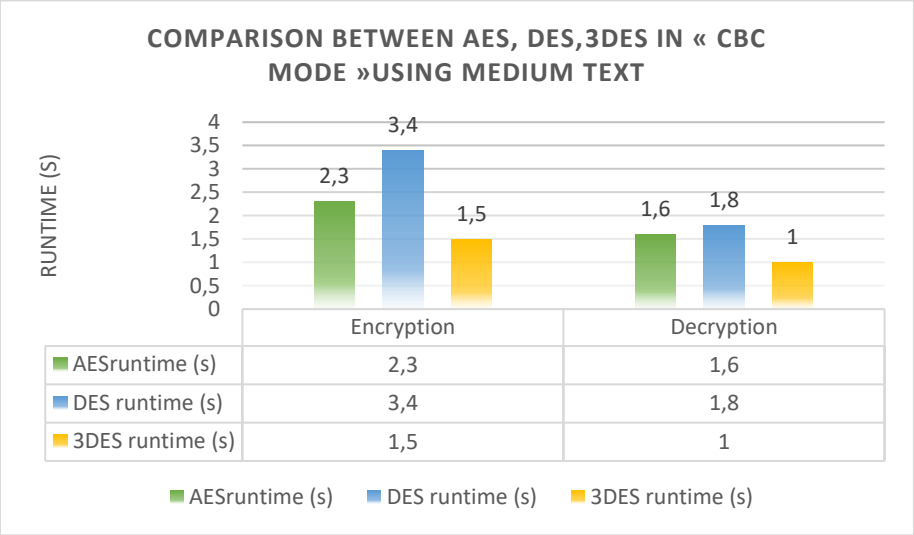


Figure 4.5-5:Histogram explain comparison between AES, DES,3DES in CBC mode using medium text

Text 4:

Table below shows the result of encryption and decryption of text 4 with the selected key, and the comparison results of the algorithms based on the previous metrics

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
AES Encryption	C6cWjT2XGOzPjeqQd...	3.90	10000000	1669743.57	39000.00	O(n^2)	Secret History: The Stor...	SIC INFORMATIQUE
AES Decryption	Secret History: The Stor...	5.20	10000000	1689153.83	52000.99	O(n^2)	C6cWjT2XGOzPjeqQd...	SIC INFORMATIQUE
DES Encryption	zyLjErkMVhqXfk/2a3T...	19.60	1250000.00	332244.76	195999.99	O(n^2)	Secret History: The Stor...	SICURITE
DES Decryption	Secret History: The Stor...	11.80	1250000.00	743050.11	118000.	O(n^2)	zyLjErkMVhqXfk/2a3T...	SICURITE
3DES Encryption	qlY 1mBmgZZBCgKu0...	11.90	1250000.00	547226.33	118999.99	O(n^2)	Secret History: The Stor...	Triple DES Algorithme M2
3DES Decryption	Secret History: The Stor...	14.00	1250000.00	626285.00	140000.00	O(n^2)	qlY 1mBmgZZBCgKu0...	Triple DES Algorithme M2

Table 4.5-8:Comparison between AES, DES,3DES in « CBC mode » using Very long text

The execution time result is representing in the following figure.

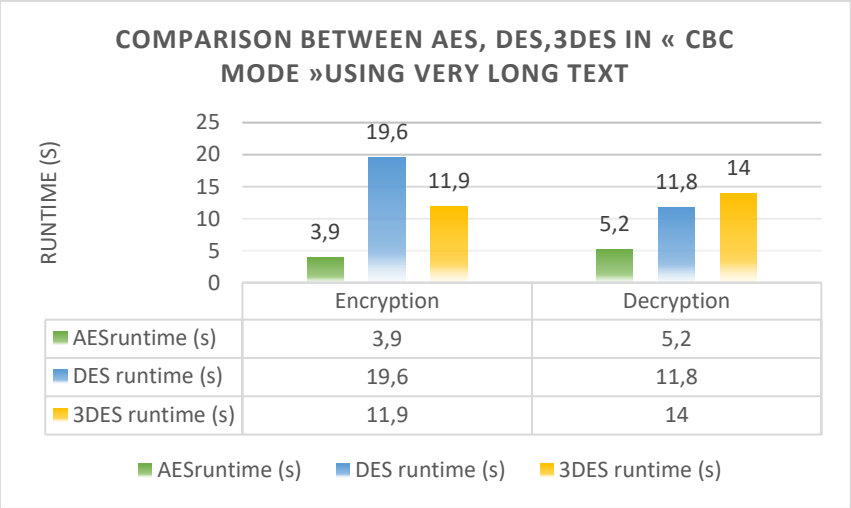


Figure 4.5-6:Histogram explain Comparison between AES, DES,3DES in CBC mode using very long text

4.5.3.1 Discussion:

CBC mode introduces some additional overhead compared to ECB mode due to the chaining process. This might lead to a slight increase in execution time and memory usage compared to ECB mode for the same underlying algorithm. However, for modern processors, this overhead is usually negligible, especially for small data sizes.

Each additional bit in the key translates to more operations the algorithm needs to perform.

4.5.3.2 Conclusion:

While ECB mode might offer slightly better raw performance for short text with short keys, its security vulnerabilities make it a poor choice for most encryption applications. CBC mode, despite a small performance overhead, provides significantly better security and is the recommended mode of operation for most block cipher algorithms like AES, DES, and 3DES.

4.5.4 Comparison between AES algorithm key size:

4.5.4.1 Comparison between AES key size in ECB mode:

the key is represented in the following table:

	128	**192**	**256**
AES Key size	SIC INFORMATIQUE	SECURITY INFORMATIQUE M2	SECURITY INFORMATIQUE M2 UNIV DZ

Table 4.5-9: keys chosen from AES algo in ECB mode

Text 5: SYMMETRIC ALGORITHMS COMPARISON

Result :

AES Encryption (128): KireBcB0YDXjG3lJ+iBR1UyHeGGx5rJLMWrFIHSsCAk=

AES Encryption (192): 48/4qMv2HJ2QAxuMNQYTJNkuLNhFhF7P4nWedH0k7RQ=

AES Encryption (256): EW8MdaA6rNQ0xvAiSpNMabECJnz8EtcRWNg//2ChhS8=

Table below shows the result of encryption and decryption of text 5 with the selected key, and the comparison results of the algorithms based on the previous metrics

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
AES 128 Encryption	KireBB0YDXjG3|J+..	2.70	10000000	11481.48	27000.00	O(n^2)	SYMMETRIC ...	SIC INFORMATIQUE
AES 128 Decryption	SYMMETRIC ...	1.00	10000000	44000.00	10000.00	O(n^2)	KireBB0YDXjG3|J+..	SIC INFORMATIQUE
AES 192 Encryption	48/4qMv2HJ2Q...	0.60	10000000	51666.66	5999.99	O(n^2)	SYMMETRIC ...	SECURITY INFORMATIQUE M2
AES 192 Decryption	SYMMETRIC ...	0.70	10000000	62857.14	6999.99	O(n^2)	48/4qMv2HJ2Q...	SECURITY INFORMATIQUE M2
AES 256 Encryption	EW8MdaA6rNQ..	1.30	10000000	23846.15	12999.99	O(n^2)	SYMMETRIC ...	SECURITY INFORMATIQUE M2 UNIV DZ
AES 256 Decryption	SYMMETRIC ...	0.60	10000000	75000.00	4999.99	O(n^2)	EW8MdaA6rNQ..	SECURITY INFORMATIQUE M2 UNIV DZ

Table 4.5-10: Comparison between AES key size in « ECB mode » using medium text

The execution time result is representing in the following figure.

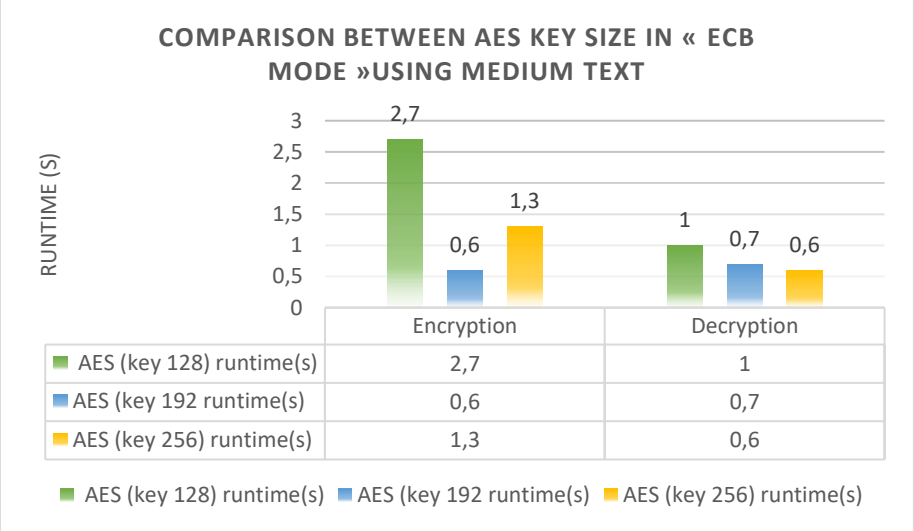


Figure 4.5-7:Histogram explain Comparison between AES key size in ECB mode using medium text

Text 6: Symmetric encryption is an encryption method where the same key is used to both encrypt and decrypt a message. Unlike asymmetric encryption, which uses a pair of keys (public and private), symmetric encryption provides faster processing speed because it only requires a single key for encryption and decryption operations.

Table below shows the result of encryption and decryption of text 6 with the selected key, and the comparison results of the algorithms based on the previous metrics

ALGORITHM Name	results	Execution time(ms)	Memory space used(octets)	Performance (caracters/s)	Energy consumption (Mj)	Algorithm complexity	Text	key
AES 128 Encryption	PMbRtaesh..	1.60	10000000	189411.48	16999.00	O(n^2)	SYMMETRIC ...	SIC INFORMATIQUE
AES 128 Decryption	SYMMETRIC ...	1.00	10000000	448000.00	10000.00	O(n^2)	PMbRtaesh..	SIC INFORMATIQUE
AES 192 Encryption	8wOFyzYUlwA...	1.50	10000000	214666.66	15000.99	O(n^2)	SYMMETRIC ...	SECURITY INFORMATIQUE M2
AES 192 Decryption	SYMMETRIC ...	1.20	10000000	373333.14	11999.99	O(n^2)	8wOFyzYUlwA...	SECURITY INFORMATIQUE M2
AES 256 Encryption	BaMgzHM71kJ...	1.10	10000000	292727.15	11000.99	O(n^2)	SYMMETRIC ...	SECURITY INFORMATIQUE M2 UNIV DZ
AES 256 Decryption	SYMMETRIC ...	1.40	10000000	320000.00	13999.99	O(n^2)	BaMgzHM71kJ...	SECURITY INFORMATIQUE M2 UNIV DZ

Table 4.5-11:Comparison between AES key size in « ECB mode »

The execution time result is representing in the following figure.

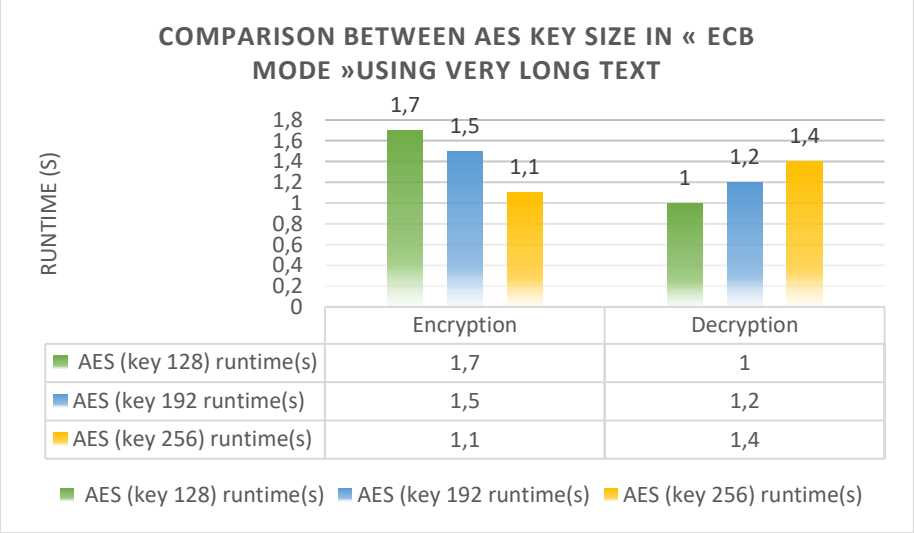


Figure 4.5-8:Histogram explain Comparison between AES key size in ECB mode using very long text

4.5.4.1.1 Observations:

The tables show a gradual increase in execution time for AES as the key size increases (from 128-bit to 192-bit and 256-bit) in medium and long text.

This is because larger keys require more complex key schedule calculations and manipulations during the encryption and decryption processes. Each additional bit in the key translates to more operations the algorithm needs to perform.

While the increase in execution time might be minimal for small data sets(text) but it can become more noticeable for larger texts.

4.5.4.1.2 Conclusion:

The tables highlight the impact of key size on the speed of AES encryption for medium and long text in ECB mode. Larger key sizes provide stronger security but come at the cost of slightly slower execution times. The specific increase in execution time will depend on the hardware used and the size of the data being encrypted.

The ideal key size depends on your specific needs. If security is paramount, a larger key size (192-bit or 256-bit) is recommended. However, if performance is a major concern and the security requirements are less stringent, a smaller key size (128-bit) might be sufficient. Most modern applications consider 128-bit AES to be secure for most purposes, but the decision ultimately depends on the specific context and risk tolerance.

4.5.4.2 Comparison between AES key size in CBC mode:

We use the same keys

Text 1: SYMMETRIC ALGORITHMS COMPARISON

Result:

Table below shows the result of encryption and decryption of text 1 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution Time(ms)	Memory space used (octets)	Performance (caractères/s)	Energy consumption (mJ)	Algorithm complexity	text	Key
Encrypt AES	L2djW3XcoOTQh4LxT...	1.000000	10000000	32000.000000	10000.000000	O(n^2)	SYMMETRIC ALGORIT...	SIC INFORMATIQUE
Decrypt AES	SYMMETRIC ALGORIT...	0.600000	10000000	108333.334410	5999.999940	O(n^2)	L2djW3XcoOTQh4LxT...	SIC INFORMATIQUE
Encrypt AES	GyZW15tbdQkrtpBloEV...	0.300000	10000000	106666.662428	3000.000119	O(n^2)	SYMMETRIC ALGORIT...	SECURITY INFORMAT...
Decrypt AES	SYMMETRIC ALGORIT...	0.500000	10000000	130000.000000	5000.000000	O(n^2)	GyZW15tbdQkrtpBloEV...	SECURITY INFORMAT...
Encrypt AES	ZjTjWdj21qh7GGJuJxY...	0.400000	10000000	79999.998808	4000.000060	O(n^2)	SYMMETRIC ALGORIT...	SECURITY INFORMAT...
Decrypt AES	SYMMETRIC ALGORIT...	0.400000	10000000	162499.997579	4000.000060	O(n^2)	ZjTjWdj21qh7GGJuJxY...	SECURITY INFORMAT...

Table 4.5-12:Comparison between AES key size in « CBC mode »

The execution time result is representing in the following figure.

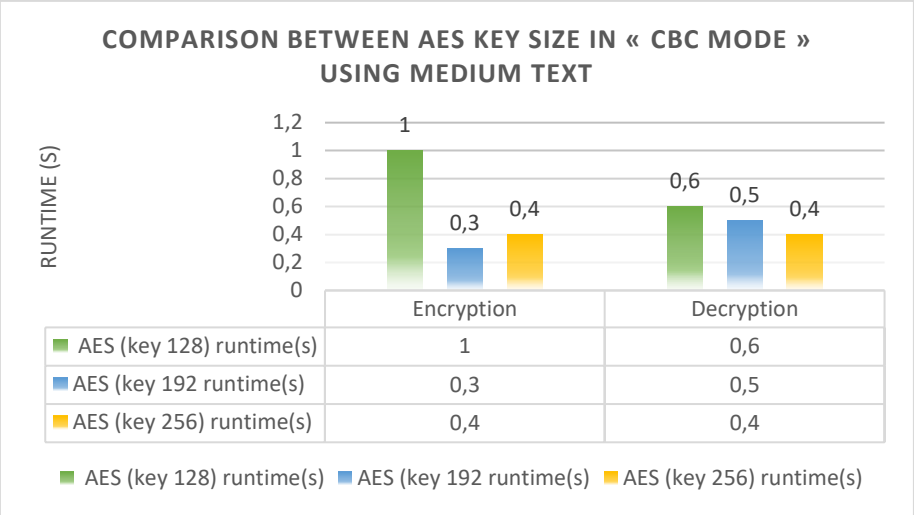


Figure 4.5-9:Histogram explain Comparison between AES key size in CBC mode using medium text

The table "COMPARISON BETWEEN AES key size "CBC mode" medium text" compares the performance of the AES encryption algorithm in CBC mode with different key sizes on medium text.

Text 6: the size of the text used is: 322 characters

Table below shows the result of encryption and decryption of text 6 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution Time(ms)	Memory space used (octets)	Performance (caractères/s)	Energy consumption (mJ)	Algorithm complexity	text	Key
Encrypt AES	mEllBswNlx0dqEXqH8...	0.700000	10000000	461428.579287	6999.999881	O(n^2)	Symmetric encryption is...	SIC INFORMATIQUE
Encrypt AES	Symmetric encryption is...	0.400000	10000000	1122499.983273	4000.000060	O(n^2)	mEllBswNlx0dqEXqH8...	SIC INFORMATIQUE
Encrypt AES	Z8qyGcEOXnHfPdLIR0...	0.800000	10000000	403749.993984	8000.000119	O(n^2)	Symmetric encryption is...	SECURITY INFORMAT...
Decrypt AES	Symmetric encryption is...	0.700000	10000000	641428.582352	6999.999881	O(n^2)	Z8qyGcEOXnHfPdLIR0...	SECURITY INFORMAT...
Chiffrement AES	zQKkMLafsCiB/PWhGs...	0.400000	10000000	807499.987967	4000.000060	O(n^2)	Symmetric encryption is...	SECURITY INFORMAT...
Decrypt AES	Symmetric encryption is...	0.900000	10000000	498888.885585	9000.000060	O(n^2)	zQKkMLafsCiB/PWhGs...	SECURITY INFORMAT...

Table 4.5-13:Comparison between AES key size in « CBC mode »

The execution time result is representing in the following figure.

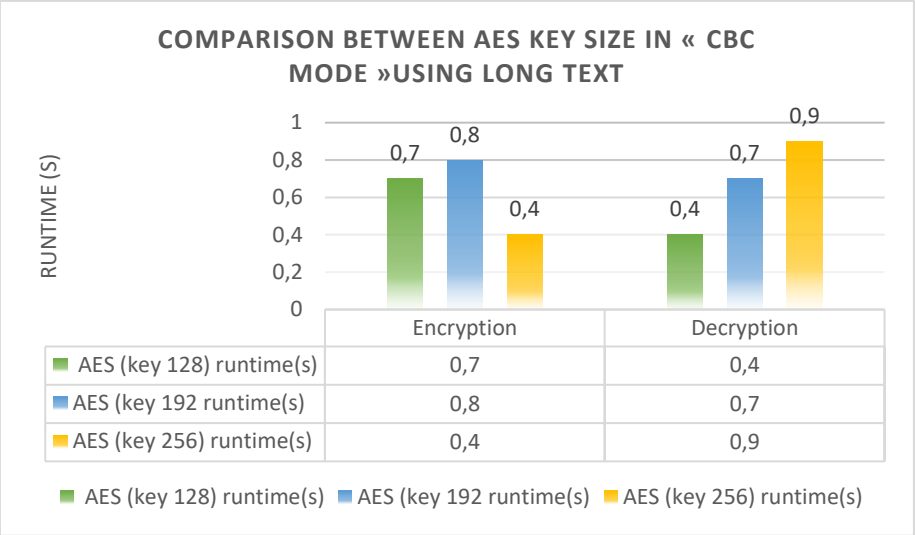


Figure 4.5-10:Histogram explain comparison between AES key size in CBC mode using long text

The table "Comparison between AES key size "CBC mode" long text" compares the performance of the AES encryption algorithm in CBC mode with different key sizes on long text.

4.5.4.2.1 Conclusion

CBC mode introduces a minor overhead compared to ECB mode due to the process of chaining ciphertext blocks. This overhead translates to a small increase in execution time.

Negligible for Modern Systems(hardware): For modern processors, especially for small to medium data sizes like the one shown in the table, the performance difference between ECB and CBC mode is usually negligible. While the increase in execution time might be minimal for small data sets(text) but it can become more noticeable for larger texts.

4.5.4.3 General conclusion for symmetric algorithms:

AES reigns supreme: Across all scenarios, AES demonstrates the best performance in terms of execution time and memory usage. It's the most efficient algorithm for all texts encryption.

DES lags behind: DES consistently exhibits the slowest performance among the three algorithms, regardless of text size, key size, or mode of operation.

3DES finds a middle ground: 3DES offers a balance between AES and DES in terms of performance. It's faster than DES but slower than AES.

Key size impact: For AES, larger key sizes (192-bit and 256-bit) provide stronger security but come at a slight cost in terms of execution speed compared to the smaller 128-bit key.
CBC mode overhead: CBC mode introduces a minor performance overhead compared to ECB mode due to the chaining process. However, for modern systems(hardware) and small to medium data sizes, this overhead is usually negligible.

4.5.4.4 Security vs. Performance Trade-off:

ECB mode - Security Flaw: While ECB mode might offer slightly better raw performance for specific cases (especially for short text), its security vulnerability makes it a poor choice. Repeated patterns in the plaintext will be reflected in the ciphertext, compromising confidentiality.
CBC mode - Security First: CBC mode addresses the security flaw of ECB and is the recommended mode of operation for most scenarios. The slight performance overhead is a small price to pay for significantly improved security.

4.5.4.5 Choosing the Right Algorithm and Mode:

Prioritize Security: For most applications, security should be the primary concern. AES in CBC mode is the best choice for both short and medium text encryption, offering excellent performance and strong security.
Performance Considerations: If raw performance is absolutely critical, and security risks are well understood and mitigated, then DES or 3DES in ECB mode might be considered (not recommended for most cases). However, for modern systems, the performance difference between ECB and CBC mode for AES is usually negligible.

4.6 Comparison between AES & RSA algorithms:

For each algorithm we use the real key size (for AES 256, RSA 512)
Generated = genre automatic by algo

Key 1: the key is represented in the following table:

Key size	256	512
AES	SIC INFORMATIC M2 CLASS 2023-24	
RSA		generated

Table 4.6-1: keys chosen from AES & RSA algos

4.6.1 Comparison between AES and RSA using medium text:

Txt2 :.

Result:

AES: yYg90lPMfErP2ToFw+ICo8zn9rWRelImLcPpSNtcvl4EfMIr46BXl4vjtoTswzXO

RSA:mvvXfKCm3j7Rx9kzuUKaazITrmEQ3Ug0x/aBxg58b3A287dipJVAOY9LPpg1xRVsN80bT3pnhxnK6k4MoLmy/w==

Table below shows the result of encryption and decryption of text 2 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution time (millisecondes)	Memory space used (octets)	Performance (caractères/s)	Energy consumption (mJ)	Algorithm complexity	text	Key
AES Encryption	LUd7qaHXU+1fZrCaLb...	0.700000	10000000	47142.857146	7000.000000	O(n^2)	ASYMMETRIC ALGORI...	SIC INFORMATIC M2 ...
AES Decryption	ASYMMETRIC ALGORI...	0.500000	10000000	130000.000000	5000.000000	O(n^2)	LUd7qaHXU+1fZrCaLb...	SIC INFORMATIC M2 ...
RSA Encryption	EKQdNf/5RSpe8vB5Cg...	2.50	10000000	13200.000000	25000.000000	O(n log n)	ASYMMETRIC ALGORI...	-----BEGIN PUBLIC KE...
RSA Decryption	ASYMMETRIC ALGORI...	4.20	10000000	20952.380953	42000.000000	O(n log n)	EKQdNf/5RSpe8vB5Cg...	-----BEGIN RSA PRIVA...

Table 4.6-2:Comparison between the algorithm AES and RSA using medium text.

The execution time result is representing in the following figure.

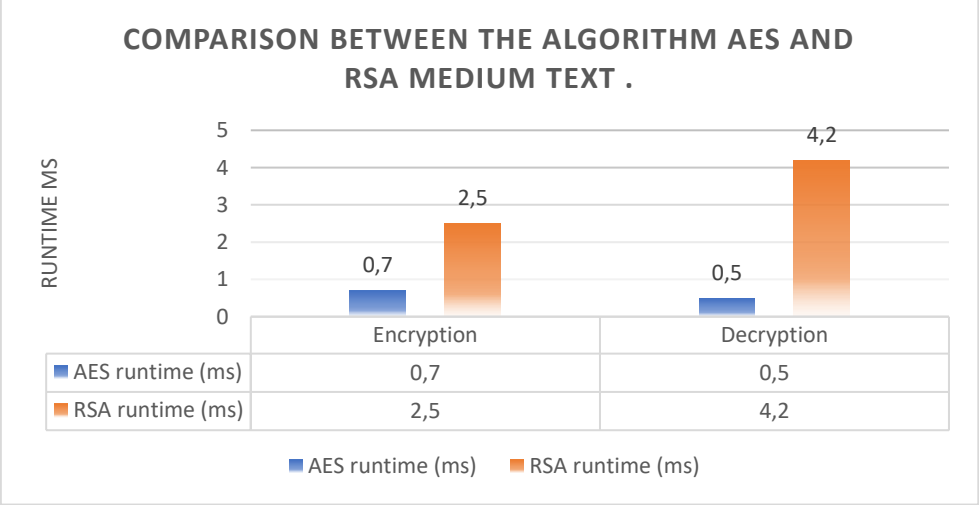


Figure 4.6-1:Histogram explain comparison between the algorithm AES and RSA using medium text.

4.6.1.1 Discussion**:**

In this table we used the key size 256 bytes (maximum size) for AES and 512 bits (minimal size) for RSA.
Encryption/Decryption Time: AES is significantly faster than RSA for both encryption and decryption. Decryption times are also faster.
Performance: AES performs better than RSA in terms of characters/ second processed.
Memory Usage: Both AES and RSA use the same amount of memory.
Energy Consumption: AES has a lower energy consumption than RSA.

4.6.1.2 Conclusion:

Based on the results in the table, AES is a better choice than RSA for encrypting and decrypting small text files. AES is significantly faster, performs better, and has a lower energy consumption than RSA.
However, it's important to remember that RSA and AES are designed for different purposes. AES is a symmetric key algorithm, which means that the same key is used for encryption and decryption. This makes it efficient for bulk encryption of data. RSA is an asymmetric key algorithm, which means that it uses a public key for encryption and a private key for decryption. This makes it more suitable for secure key exchange and digital signatures.
So, the best choice for you will depend on your specific needs. If you need to encrypt and decrypt small text files quickly and efficiently, then AES is a good option. If you need to securely exchange keys or create digital signatures, then RSA is a better choice.

4.6.2 Comparison between AES and RSA using text size 501 bits:

Text 7: the size of the text used is: 501 characters

the key is represented in the following table:

Key size	256	4092
AES	SIC INFORMATIQUE	
RSA		generated

Table 4.6-3: keys chosen from AES & RSA algos

Table below shows the result of encryption and decryption of text 7 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution time (millisecondes)	Memory space used (octets)	Performance (caractères/s)	Energy consumption (mJ)	Algorithm complexity	text	Key
Encrypt AES	ZALOuY6gTOgw8SFE+...	2.900000	10000000	172758.620667	29000.000004	O(n^2)	Secret History: The Stor...	SIC INFORMATIQU
Decrypt AES	Secret History: The Stor...	2.200000	10000000	310909.090751	22000.000011	O(n^2)	ZALOuY6gTOgw8SFE+...	SIC INFORMATIQU
RSA Encryption	MW0uQRinWNY11tgxr...	18.50	10000000	27081.081081	185000.000000	O(n log n)	Secret History: The Stor...	-----BEGIN PUBLIC
RSA Decryption	Secret History: The Stor...	178.50	10000000	3831.932773	1785000.000000	O(n log n)	MW0uQRinWNY11tgxr...	-----BEGIN RSA PRI

Table 4.6-4:Comparison between algorithms AES and RSA using txt size 501 bits

The execution time result is representing in the following figure.

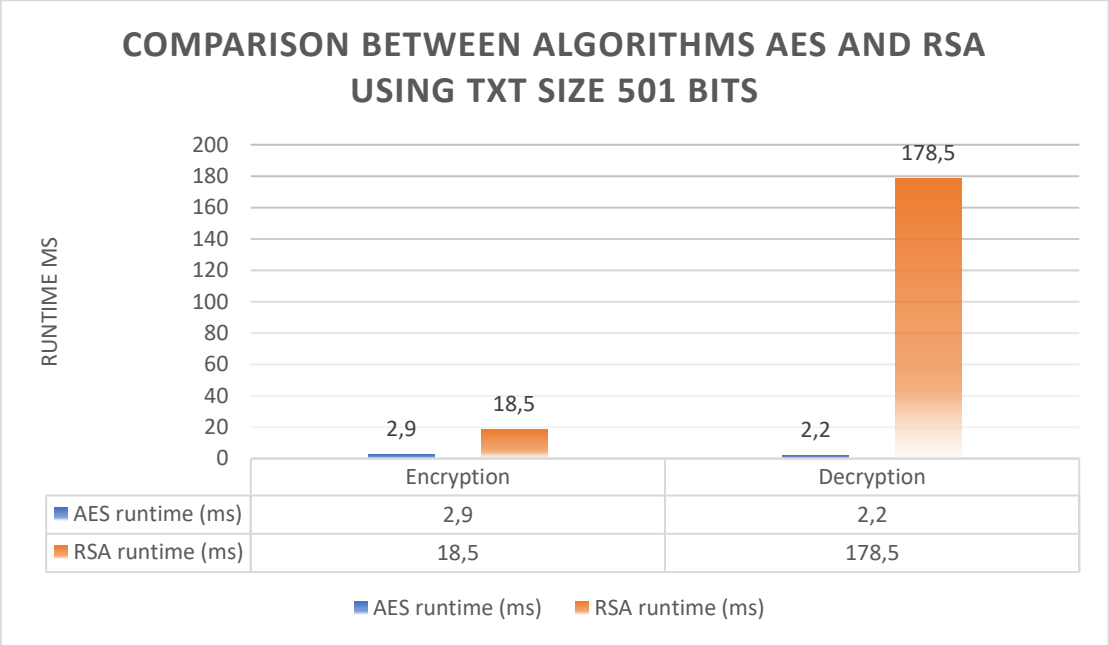


Figure 4.6-2:Histogram explain comparison between algorithms AES and RSA using txt size 501 bits

4.6.2.1 Conclusion:

In these last table we have almost the same results except the we see a big increase in the execution time that of RSA due to the uses a more complex mathematical structure and the large key size (4092 bits). This increased security often comes at the cost of speed.

4.7 RSA key size comparison:

Due to the nature of its mathematical foundations, the RSA algorithm has limitations on the size of the input data that can be directly encrypted and decrypted. These

limitations are related to the key size used in the algorithm. Here's a brief overview of how it works:

4.7.1 Key Size and Maximum Input Length:

Key Size: RSA keys come in different sizes, commonly 512, 1024, 2048, 3072 or 4096 bits. The maximum length of data that can be encrypted directly is slightly less than the key size. This is because the RSA algorithm adds padding to the plaintext to ensure security.

4.7.2 Maximum Input Length example calculations:

PKCS#1 v1.5 Padding: Typically adds 11 bytes of padding. the 11 bytes of padding refer to the overhead added to the plaintext to ensure secure encryption.

Rule: Maximum plaintext size=Key size in bytes−11

- 512 bit key (64 bytes): With PKCS#1 v1.5 padding:

Maximum plaintext length = 64 - 11 = 53 bytes.

- 1024 bit key (128 bytes): With PKCS#1 v1.5 padding:

Maximum plaintext length = 128 - 11 = 117 bytes.

- 2048 bit key (256 bytes): With PKCS#1 v1.5 padding:

Maximum plaintext length = 256 - 11 = 245 bytes.

- 3072 bit key (384 bytes): With PKCS#1 v1.5 padding

Maximum plaintext size = 384 bytes - 11 = 373 bytes.

- 4096 bit key (512 bytes): With PKCS#1 v1.5 padding:

Maximum plaintext length = 512 – 11 = 501 bytes.

Here we use a medium txt with key generated automatically

Table below shows the result of encryption and decryption of text 2 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution time (millisecondes)	Memory space used (octets)	Performance (caractères/s)	Energy consumption (mJ)	Algorithm complexity	text
RSA Encryption KEY: 512	RP3rGmaU…	2.70	10000000	11851.851855	26999.999993	O(n log n)	SYMMETRIC ALGORITHMS COMPARISON
RSA Decryption KEY: 512	SYMMETRIC ALGORITHMS COMPARISON	4.00	10000000	22000.000000	40000.000000	O(n log n)	RP3rGmaUAO0MFt…
RSA Encryption KEY:1024	JEsOMiekLkFe/…	1.20	10000000	26666.666683	11999.999993	O(n log n)	SYMMETRIC ALGORITHMS COMPARISON
RSA Decryption KEY:1024	SYMMETRIC ALGORITHMS COMPARISON	21.10	10000000	8151.658768	211000.000006	O(n log n)	JEsOMiekLkFe/AHp2…
RSA Encryption KEY:2048	Qb/5E37nG…	3.70	10000000	8648.648648	37000.000002	O(n log n)	SYMMETRIC ALGORITHMS COMPARISON
RSA Decryption KEY:2048	SYMMETRIC ALGORITHMS COMPARISON	33.90	10000000	10147.492625	339000.000004	O(n log n)	Qb/5E37nGu…
RSA Encryption KEY: 3072	cGc9zUlmAk+…	10.60	10000000	3018.867924	106000.000006	O(n log n)	SYMMETRIC ALGORITHMS COMPARISON
RSA Decryption KEY: 3072	SYMMETRIC ALGORITHMS COMPARISON	155.90	10000000	3284.156511	1559000.000004	O(n log n)	cGc9zUlmAk+…
RSA Encryption KEY: 4096	Jken36M2W4XJZ…	14.60	10000000	2191.780822	146000.000006	O(n log n)	SYMMETRIC ALGORITHMS COMPARISON
RSA Decryption KEY: 4096	SYMMETRIC ALGORITHMS COMPARISON	190.60	10000000	3588.667366	1905999.999996	O(n log n)	Jken36M2W4XJ…

Table 4.7-1:Comparison between RSA key size in « ECB mode » using text 5

The execution time result is representing in the following figure.

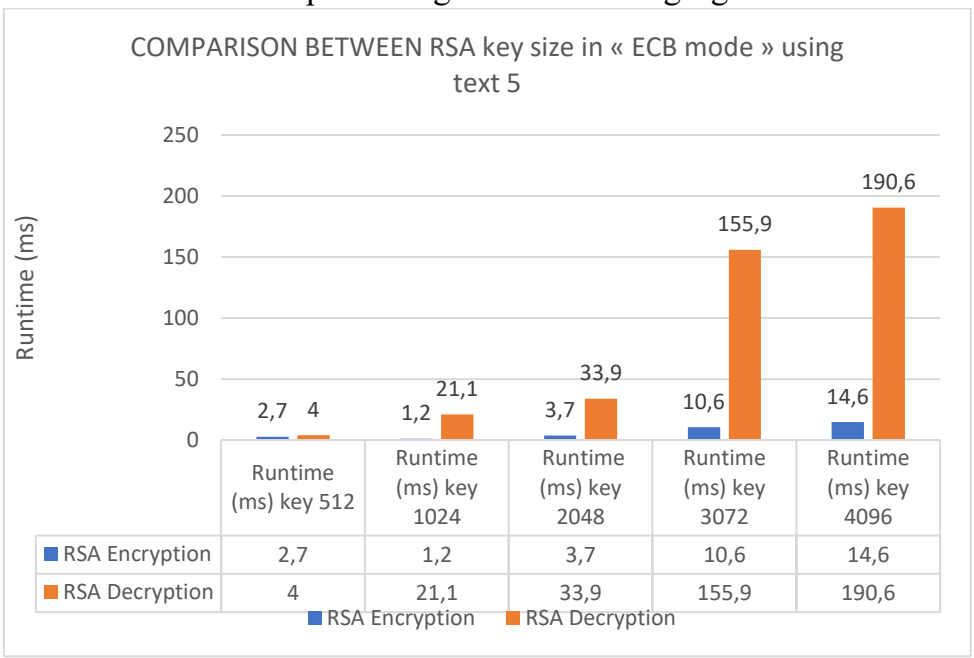


Figure 4.7-1;Histogram explain comparison between RSA key size in « ECB mode » using text 5

4.7.3 Discussion:

Execution Time: There's a significant trade-off between security and performance. As the key size increases, the execution time for both encryption and decryption increase dramatically. This is because RSA relies on complex mathematical operations involving very large numbers. These operations take longer to compute with larger keys.

Performance: Performance, measured in characters/second processed, decreases with increasing key size. This is a direct consequence of the longer execution times. Encrypting and decrypting data becomes slower as the key size grows.
Memory Usage: Fortunately, memory usage remains relatively constant for RSA regardless of the key size. The encryption and decryption processes primarily involve manipulating large numbers, which doesn't require significant additional memory allocation.
Energy Consumption: Energy consumption increases as the key size increases. The complex mathematical operations involved in RSA become more energy-intensive with larger keys. This can be a concern for battery-powered devices or environments where energy efficiency is a priority.

4.7.4 Conclusion:

Larger key sizes offer better security but come at the cost of slower performance, higher energy consumption. For small text files, the performance penalty might outweigh the security benefit of a larger key.
Choose the key size based on your specific needs: prioritize speed for small files, prioritize security for large data volumes. Current recommendations suggest a minimum key size of 2048 bits for RSA. This offers a good balance between security and performance for most use cases.

4.8 Hache Algorithm Comparison:

Hashing algorithms are not intended for encryption and decryption of text with a key. They work unidirectionally, transforming an input (plain text) into an output (digital fingerprint) called a “hash” or “digest”. Hashing is a unidirectional and non-reversible process, meaning that once data has been hashed, it is not possible to recover the original data from the hash Depending on the context and security requirements, hashing algorithms may or may not use a key. For simple hashing of the data, no key. is required. In this section of the chapter, we will experiment with texts of different lengths without resorting to using a key to compare hashing algorithms.

4.8.1 Comparison between Hashing algorithms using medium text:

Txt 2:
Result:
MD5: 4ec602277613706086c8a76becc5f80d

SHA3:
984ae7505631257c078701d56e9fdfb7dcba087447c18dd7e0cecb4444041d249d874e53b22532e1740b49489b41caa8d7bb67e48a65359ba076e1870df958ab
SHA256: f254a429ea92d0600c6f7642d632f63e6a24cc65e1804512a781d65e4b069be1
Table below shows the result of encryption and decryption of text 2 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution time	Memory space used	Energy consumption	Performance	Complexity of the algorithm
MD5	4ec602277613706086c...	0.5999999940395355 ms	9.54 MB	6000.00 mJ	63.33 caractères/millisecondes	O(38)
SHA3	984ae7505631257c078...	1.2999999821186066 ms	9.54 MB	13000.00 mJ	29.23 caractères/millisecondes	O(38)
SHA256	f254a429ea92d0600c6f...	0.30000001192092896 ms	9.54 MB	3000.00 mJ	126.67 caractères/millisecondes	O(38)

Table 4.8-1:Comparison between hashing algorithms using medium text

The execution time result is representing in the following figure.

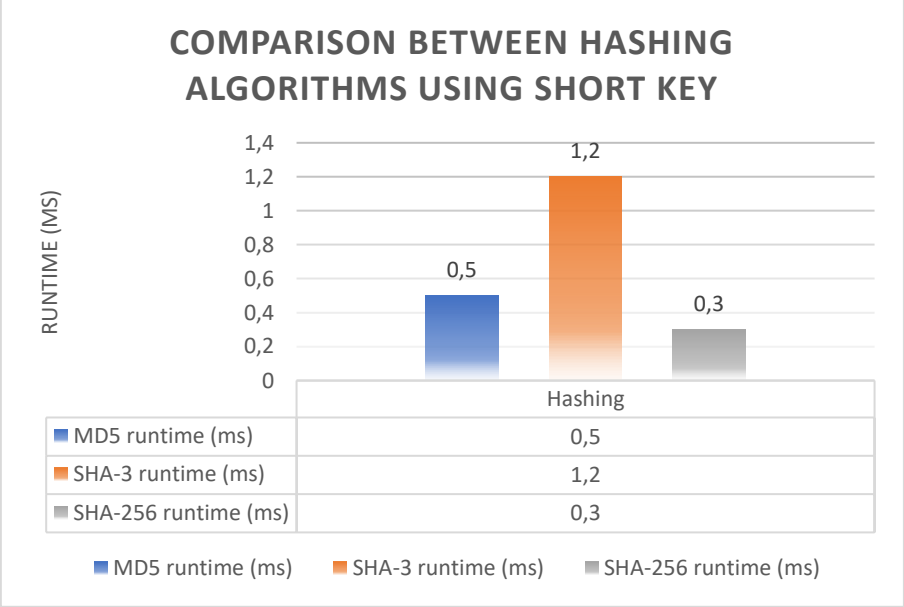


Figure 4.8-1: Histogram explain comparison between hashing algorithms using medium text

4.8.1.1 Observations

- SHA-3 has the slowest execution time compared to MD5 and SHA-256.
- All three algorithms use approximately the same amount of memory space.
- MD5 has the highest energy consumption followed by SHA-3 and SHA-256.
- SHA-256 has the best overall performance followed by MD5 and SHA-3.

4.8.1.2 Conclusion

- SHA-256 appears to be the most efficient algorithm among the three based on this table. It has the fastest execution time and lowest energy consumption while using the same amount of memory space as the other two algorithms.
- It is important to note that this table only compares the performance of these three algorithms on medium text. The relative performance of these algorithms could vary depending on the size and type of data being hashed.

4.8.2 Comparison between Hashing algorithms using very long text:

Txt 4:

Result:

MD5: 31b72f039c39b3f2e0bef78f08c04235

SHA3:

eb92b78a3bd4514e789626965075400fd12e44da6639c06a3d9512f5f7007648173edf018ceb83df07b98b386d46986027d6a0cd38e4b0cd907f581f90f224ca

SHA256:

ebc55e2cbd5c94bbf8905c5fdc6c84f0383f2ff99037e29ca0fce6110821c2a9

Table below shows the result of encryption and decryption of text 4 with the selected key, and the comparison results of the algorithms based on the previous metrics

Algorithm name	Result	Execution time(ms)	Memory space used(MB)	Energy consumption(mJ)	Performance (caractères/millisecondes)	Complexity of the algorithm
MD5	31b72f039c39b3f2e0be...	1.9000000059604645	9.54	19000.00	3427.37	O(n)
SHA3	eb92b78a3bd4514e789...	16.5	9.54	165000.00	394.67	O(n)
SHA256	ebc55e2cbd5c94bbf890...	1.2000000029802322	9.54	12000.00	5426.67	O(n)

Table 4.8-2:Comparison between Hashing algorithms using very long text

The execution time result is representing in the following figure.

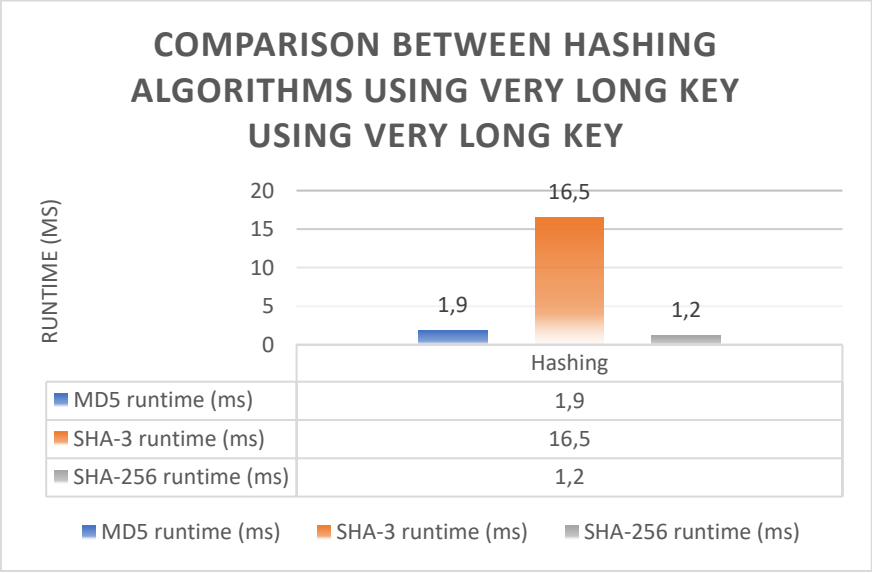


Figure 4.8-2:Histogram explain Comparison between Hashing algorithms using very long Key using very long text

4.8.2.1 Observation:

We obtain almost the same results with a small deferent in output value of the metrics. As we said with the medium text, SHA-3 is the slowest algorithm of the three.SHA-256 is the fastest algorithm of the three, SHA-3 is the best performing algorithm of the three. It has the same average memory consumption as MD5 and SHA-3 and the same complexity.

4.8.2.2 Conclusion:

In these last table we have almost the same results except the we see a big increase in the execution time that of SHA-3 due to the uses a more complex mathematical structure compared to SHA-256. SHA-3 involves more processing rounds compared to SHA-256 to achieve its higher level of security. Each round adds to the overall execution time. This increased security often comes at the cost of speed.

4.8.3 Hash tables analyses:

When choosing a hashing algorithm, consider both data size and your specific needs:

- For smaller datasets: If speed is crucial, SHA-256 might be preferable, especially if security offered by SHA-256 is sufficient for your application.
- For very large datasets: The performance gap between SHA-256 and SHA-3 might narrow. Here, the choice could depend on:
- Security requirements: If paramount, SHA-3 might be a better option despite potentially slower speed.
- Processing power constraints: If processing power is limited, SHA-256 might be a more practical choice, even for large datasets.
- Overall, understanding the trade-off between speed and security, as well as the impact of data size, is crucial for selecting the most suitable hashing algorithm for your application.

4.9 Conclusion:

We analyzed and compared several cryptographic algorithms in this research, with particular focus on hashing, symmetric, asymmetric, and classic cryptography. Each type of algorithm has unique properties designed to meet particular performance and security requirements. The best cryptographic algorithm to utilize will depend on the particular needs of the user. Asymmetric algorithms are necessary for safe exchanges and public key management, while symmetric algorithms work well in situations needing high performance and centralized key management. Hash functions are still necessary to guarantee the authenticity and integrity of data. Considering the advantages and disadvantages of every algorithm, decision-makers, or users, can choose the cryptographic algorithms that are most appropriate for their particular requirements.

General Conclusion

In conclusion, this dissertation provides a holistic view of computer security, from its fundamental principles to advanced cryptographic techniques and system architecture.

Experimentation and evaluation of cryptographic algorithms form a significant part of the study. Various algorithms are compared based on performance, execution time, memory used, energy consumption, and complexity for different applications. The comparisons cover both classical and modern algorithms. Cryptography, a cornerstone of computer security, is explored in depth. Classical cryptographic methods such as Caesar, Playfair, and Vigenère ciphers are reviewed, followed by an extensive analysis of modern cryptographic techniques including symmetric algorithms (AES, DES, 3DES) and asymmetric algorithms (RSA), as well as hash functions (MD5, SHA-3, SHA-256).

To further advance the field of computer security, especially in cryptographic algorithm comparison, we want to focus on the following future directions: incorporating comparisons of the Elliptic Curve Cryptography (ECC) algorithm, which presents unique challenges and opportunities due to its efficiency and security features. Additionally, we aim to study the cryptanalysis of each algorithm and create a website dedicated to cryptanalysis, algorithm robustness, and techniques for key breaking to evaluate the extent of each algorithm's resistance.

Bibliography

[1]: Thawte (2013),Histoire du chiffrement et de ses méthodes .Retrieve it from Thawte.fr : history-cryptography.pdf (thawte.fr)

[2] : Cisco Systems(2024), What Is IT Security? .Retrieve it from Cisco.com : http://www.cisco.com/c/en/us/products/security/what-is-it-security.html

[3] : Cryptographie, Dr. BENIDRIS Fatima zohra (Juin 2020)Cryptographie..Retrieve it from : e-biblio.univ-mosta: http://ebiblio.univmosta.dz/bitstream/handle/123456789/19428/PINF05.pdf?isAllowed=y&sequence=1

[4] :Geeksforgeeks(22 May, 2023), Advanced Encryption Standard (AES)..Retrieve it from geeksforgeeks.org : https://www.geeksforgeeks.org/advanced-encryption-standard-aes/?fbclid=IwAR3mbrp-JMJurYWsE31pchjOKCEFOHdDQ0utBismyAIKu1hBEEHTP6wdRds_aem_AftKg2a2YuCISBI1kUPyNBqLP23CYykRAj_8uLidjYFlVW8kZ5B5WqyGM-_K1fj4sVYYeb1TFdME7e6VWVAzDZ5e

[5] : Scaler Topics (2024)Monoalphabetic Cipher. Retrieve it from scaler.com https://www.scaler.com/topics/monoalphabetic-cipher/

[6] :Comparitech (2018)What is 3DES encryption and how does DES work?Retrieve it from comparitech.com: https://www.comparitech.com/blog/information-security/3des encryption/#The_history_of_3DES_encryption

[7]: Geeksforgeeks, (07-04- 2023), What is Information Security? Retrieve it from: geeksforgeeks.org https://www.geeksforgeeks.org/what-is-information-security/

[8] : **G**eeksforGeeks, (08-05-2024). what-is-monoalphabetic cipher. Retrieve it from: geeksforgeeks.org https://www.geeksforgeeks.org/what-is-monoalphabetic-cipher/

[9] : Département d'informatique Université de Batna 2-Dr. Khaled HAMOUID-Cryptographie Retrieve it from staff.univ-batna2.dz : https://staff.univ-batna2.dz/sites/default/files/hamouid_khaled/files/crypto1-2022.pdf

[10]: Planetcalc, Caesar cipher Retrieve it from: planetcaljournc.com, https://planetcalc.com/1434/

[11]: Planetcalc, Chiffre de Vigenère. Retrieve it from : planetcalc.com
https://fr.planetcalc.com/2468/

[12] : Geeksforgeeks(13 Apr, 2023),Playfair Cipher with Examples.Retrieve it from geeksforgeeks.org :
https://www.geeksforgeeks.org/playfaircipherwithexamples/?ref=gcse&fbclid=IwAR2E5RCyABM1SRPrdAeLc7cvsh0NWTc8lbVEeyTj1JdsVZL0OukebftSMs

[13]: ResearchGate .Retrieve it from: researchgate.net https://www.researchgate.net/figure/Block-Diagram-showing-working-of-DES-DES-algorithm-consists-of-the-following-steps_fig1_324045946

[14]: Simplilearn (29-05-2024). RSA Algorithm: Secure Your Data with Public-Key Encryption. Retrieve it from: simplilearn.com https://www.simplilearn.com/tutorials/cryptography-tutorial/rsa-algorithm

[15] Silicon(2024),Qu'est-ce que le chiffrement asymétrique ?Retrieve it from : fiches-pratiques.silicon.fr

https://fiches-pratiques.silicon.fr/Thematique/cybersecurite-1338/FichePratique/Tout-savoir-sur-le-chiffrementasymetrique365810.htm?fbclid=IwZXh0bgNhZW0CMTAAAR0qxHIilUa7BJanMWMIFpsaMfIoj_K5khzEm5lSsYb3Ry6E7R309r_QA_aem_AZIF849be5UBDm4UTFFcr7j1CUWvUhbJfErKqxum65SyKS05ssbQDhRg8AWXhsd1npqklrl8fx_k_va-VxEhK

[16] Lotmani Z., Ismail E. Y.(2013). *de l'Université de Mostaganem* (Master dissertation, Université Abdelhamid Ibn Badis Mostaganem)..Retrieve it from : e-biblio.univ-mosta.dz
http://e-biblio.univ-mosta.dz/bitstream/handle/123456789/9370/MINF26.pdf?sequence=1&isAllowed=y&fbclid=IwZXh0bgNhZW0CMTAAAR1Vx3PBOq1tVsc2GqTXqXNLREzW8BuZTL3oavoLpLXcQzwcSMAQEGTkGR0_aem_ATqNyTlGTwiZceCGrZ8Mydzzd-346MGVJRf3nVoFrPrIHvMBHuzJE8HbnKg7rCz6fbfFNwuUhNNgkPQyJLIlCk6-

[17] lemagit-TechTarget- (2007 - 2024),RSA (algorithme) .Retrieve it from : lemagit.fr
https://www.lemagit.fr/definition/RSA-algorithme

[18] Simplilearn (09-02-2024). What-is-des. Retrieve it from: simplilearn.com
https://www.simplilearn.com/what-is-des-article#initial_permutation_ip

[19] Medium(2023),TripleDES(3DES)Encryption Features,Process,Advantages,and Applications Retrieve it from :medium.com
https://cyberw1ng.medium.com/triple-des-3des-encryption-features-process-advantages-and-applications-2023-587e5a092789

[20]Simplilearn,Shruti M (Oct 20, 2023) Message-Digest Algorithm 5: Overview and How Does it Work? Retrieve it from :simplilearn.com
https://www.simplilearn.com/tutorials/cyber-security-tutorial/md5-algorithm#what_is_the_md5_algorithm

[21] Pittalia, P. P. (2019). A comparative study of hash algorithms in cryptography. *International Journal of Computer Science and Mobile Computing*, *8*(6), 147-152.

[22] "Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family [U.S. Federal Register Vol. 72 No. 212)]" (PDF). November 2, 2007. Archived (PDF) from the original on March 31, 2011. Retrieved July 18, 2017.

[23] PIERRE, BARTHELEMY, ROBERT, ROLLAND, et PASCAL, VERON. *Cryptographie: principes et mises en œuvre/2ème édition revue et augmentée*. Lavoisier, 2012.

[24] Bitcoinwiki,(December 21, 2023).SHA-3 ,Retrieve it from : bitcoinwiki.org https://bitcoinwiki.org/wiki/sha-3#References

[25]Simplilearn, Baivab K. J. ,(Aug 29, 2023).A Definitive Guide to Learn The SHA-256.Retrieve it from: simplilearn.com https://www.simplilearn.com/tutorials/cyber-security-tutorial/sha-256-algorithm#what_is_the_sha256_algorithm

[26] Planetcalc.Chiffre de Playfair Retrieve it from : planetcalc.com https://fr.planetcalc.com/7751/

[27] W3schools ,JavaScript Tutorial .Retrieved from:w3schools.com, https://www.w3schools.com/js/default.asp

[28] Code, Visual Studio.Learn to code with Visual Studio Code .Retrieved from: code.visualstudio https://code.visualstudio.com/docs

[29] W3schools ,Bootstrap Tutorial .Retrieved from:w3schools.com, https://www.w3schools.com/bootstrap/default.asp

[30] W3schools ,HTML Tutorial .Retrieved from:w3schools.com, https://www.w3schools.com/html/default.asp

[31] W3schools ,CSS Tutorial .Retrieved from:w3schools.com, https://www.w3schools.com/css/default.asp

[32] CryptoJS .Retrieved from: cryptojs.gitbook.io ,https://cryptojs.gitbook.io/docs

[33] the-x.cn(2022), AES encryption / decryption. Retrieve it from: https://the-x.cn/en-US/cryptography/Aes.aspx

[34] the-x.cn (2022), DES encryption/decryption. Retrieve it from: https://the-x.cn/en-US/cryptography/Des.aspx

[35] devglan, Triple DES Encryption and Decryption Online Tool. Retrieve it from: devglan.com https://www.devglan.com/online-tools/triple-des-encrypt-decrypt

[36] jsencrypt, Online RSA Key Generator. Retrieve it from: travistidwell.com https://travistidwell.com/jsencrypt/demo/

[37] md5calc, Online MD5 Hash Calculator. Retrieve it from: md5calc.com https://md5calc.com/hash#google_vignette

[38] md5calc, Online SHA3-256 Hash Calculator. Retrieve it from: md5calc.com, https://md5calc.com/hash/sha3-256

[39] xorbin, SHA-256 hash calculator. Retrieve it from: xorbin.com, https://xorbin.com/tools/sha256-hash-calculator