People's Democratic Republic of Algeria

الجمهورية الجزائرية الديمقراطية الشعبية

Ministry of Higher Education and Scientific Research

وزارة التعليم العالي و البحث العلمي

Yahia Fares University of Medea

جامعة يحيى فارس المدية



Thesis

Submitted for the degree Master in Computer Science

Option: Systems engineering and web technology

Implementation of a hybrid cryptosystem ECC-AES

Presented by:
Mr. LADJLET Mourad

Supervisor:
Mr. CHETOUANE Mohamed

Defended on 24 June 2024, In front of the composed jury
Of : Mr. MESSAOUD Rahim
Mr. TIBERKAK Allal

Class of: 2023/2024

Thanksgiving

We express our heartfelt gratitude to God for granting us help, patience, and the courage to carry out this work while maintaining our health throughout this year of study.

I would like to warmly thank our supervisor, CHETOUANE Mohamed, for his invaluable assistance and the knowledge he has shared with us. I am deeply grateful for his availability and the quality of his advice.

I extend my thanks to the members of the jury for their interest in this work and for accepting to be our examiners.

I take this opportunity to express our profound gratitude to the university professors, as well as the managers and staff, for their sincere and constant efforts.

A special thanks to my parents for their love, advice, and unconditional support, both morally and financially. I am also grateful to my brothers, sisters, and entire family, who have enabled us to pursue my studies and, consequently, to complete this thesis.

I would like to express my gratitude to my friends and colleagues for their moral and intellectual support throughout our journey.

Finally, I thank all those who, whether near or far, have Supported me in carrying out this work.

Abstract

This project aims to create an innovative hybrid cryptosystem by leveraging the strengths of both symmetric and asymmetric encryption techniques. Our approach combines the speed of the Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode for data encryption with the robust security of Elliptic Curve Cryptography (ECC), specifically the NIST P-256 curve, for secure key exchange using the Elliptic Curve Diffie-Hellman (ECDH) protocol . Our cryptosystem overcomes the limitations of current cryptographic systems by offering both speed and security.

Comprehensive tests have been carried out to evaluate the performance of our hybrid system compared to other cryptographic approaches. These tests measured the execution time of encryption and decryption processes on different data sizes, as well as the time needed to generate keys. The results have demonstrated the remarkable efficiency of our cryptosystem in terms of speed. In order to validate the proposed system in terms of security, extensive tests have been carried out to assess its resilience to various types of attacks.

Key words:

Cryptography, Cryptosystem, AES, ECC,ECDH.

Résumé

Ce projet vise à créer un cryptosystème hybride innovant en tirant parti des avantages des techniques de chiffrement symétrique et asymétrique. Notre approche combine la vitesse de l'Advanced Encryption Standard (AES) en mode de chiffrement par blocs (CBC) pour le cryptage des données avec la sécurité robuste de la cryptographie du courbe elliptique (ECC), en particulier la courbe NIST P-256 pour l'échange sécurisé des clés en utilisant le protocole Elliptic Curve Diffie-Hellman (ECDH). Notre cryptosystème surmonte les limites des systèmes cryptographiques actuels en offrant à la fois la vitesse et la sécurité.

Des tests complets ont été effectués pour évaluer les performances de notre système hybride par rapport à d'autres approches cryptographiques. Ces tests ont mesuré les temps d'exécution des processus de cryptage et de décryptage sur différentes tailles de données, ainsi que les temps nécessaires à la génération de clés. Les résultats ont démontré l'efficacité remarquables de notre cryptosystème en termes de vitesse. Pour valider le système proposé en terme de sécurité, des essais approfondis ont été effectués pour évaluer sa résilience à divers types d'attaques.

Mots clés :
Cryptographie, Cryptosysteme, AES, ECC, ECDH.

Contents

1 Generalities on Cryptography

1.1 Basic Cryptography Vocabulary . . . 10
1.2 Goals of cryptography . . . 10
1.3 Types of cryptography . . . 11
1.3.1 Classic cryptography . . . 11
1.3.2 Modern cryptography . . . 13
1.4 Conclusion . . . 18

2 State of the art on AES and ECC

2.1 AES (Advanced Encryption Standard) . . . 19
2.1.1 Presentation of the AES . . . 19
2.1.2 Principle of AES . . . 20
2.1.3 AES transform functions . . . 20
2.1.4 Advantages and Disadvantages of AES . . . 25
2.1.5 AES Encryption/Decryption With CBC . . . 26
2.2 Elliptic Curve Cryptography (ECC) . . . 27
2.2.1 Using cryptography based on elliptic curves . . . 27
2.2.2 General . . . 29
2.2.3 Overview of elliptic curves . . . 30
2.2.4 Operations on Elliptic Curve Points . . . 32
2.2.6 Generating keys (NIST P-256 curve) . . . 35
2.2.7 Key Exchange Protocol(Diffie-Hellman ECDH) . . . 36
2.3 Hybrid System . . . 37
2.4 Conclusion . . . 37

3 Our contribution

3.1 Our method . . . 38
3.1.1 Algorithm Selection . . . 38
3.2 Cryptosystem Design . . . 39
3.2.1 Main Key Exchange . . . 39
3.2.2 AES Encryption. . . . 39
3.2.3 AES Decryption. . . . 40

3.3 Example 41
3.3.1 Key Generation and Exchange 41
3.3.2 AES Encryption 43
3.3.3 AES Decryption 45
3.4 Conclusion 47

4 Implementation and Tests

4.1 Development Environment 48
4.2 Implementation results according to execution time 49
4.3 Comparative Analysis 50
4.3.1 Analysis 51
4.4 Discussion on security against attacks 51
4.4.1 Brute Force Attack : 51
4.4.2 Man in the midlle 52
4.5 Our implementation interface 53
4.6 Conclusion 55

List of Figures

1.1 Caesar cipher: . . . 28
1.2 Vigenère table. . . . 28
1.3 Diagram illustrates Block encryption. . . . 30
1.4 Diagram illustrates symmetric encryption. . . . 30
1.5 Diagram illustrates asymmetric encryption. . . . 32
2.1 AES Encryption and Decryption. . . . 39
2.2 SubBytes/InvSubBytes transformation. . . . 40
2.3 S-Box transformation table. . . . 40
2.4 ShiftRows transformation. . . . 41
2.5 MixColumns Transformation. . . . 41
2.6 InvMixColumns Transformation. . . . 42
2.7 Round Key transformation. . . . 42
2.8 Overall key expansion algorithm. . . . 43
2.9 CBC-mode encryption and decryptio. . . . 44
2.10 Secp256k1 curve used by Bitcoin. . . . 46
2.11 Examples of elliptic curves. . . . 49
2.12 Addition Points. . . . 51
2.13 Doubling of points. . . . 52
2.14 Example of ECC version of Diffie Hellman Protocol. . . . 54
3.1 Encryption and decryption algorithm. . . . 61
4.1 Encryption and Decryption times for different message sizes. . . . 72
4.2 Time needed to break AES in CBC mode via brute force attack. . . . 73
4.3 Man-in-the-Middle Attack on Di ie-Hellman. . . . 74
4.4 Full Interface. . . . 75
4.5 key's and CBC IV Geneartion section. . . . 75
4.6 Encryption Section. . . . 76
4.7 Decryption Sectionn. . . . 77

List of Tables

1.1 Advantages and Disadvantages of Symmetric Encryption 31
1.2 Advantages and Disadvantages of Asymmetric Encryption. 33
2.1 Key size comparison between RSA and ECC 46
3.1 Example of CBC Encryption 63
4.1 Number of characters and time required to encrypt and decrypt 71
4.2 Performance of AES-256 and ECC-256 Hybrid System 72
4.3 Performance of RSA and AES Hybrid System 72
4.4 Performance of Di ie-Hellman and AES Hybrid System 73

List of Acronyms

AES Advanced Encryption Standard.

ECC Elliptic Curve Cryptography.

RSA Rivest Shamir Adleman.

PGP Pretty Good Privacy.

GnuPG GNU's Not Unix Privacy Guard.

CBC Cipher Block Chaining.

ECDH Elliptic Curve Diffie Hellman.

General Introduction

The development of networks and the Internet has greatly boosted electronic commerce and cryptology. Since information is transferred across these networks in all fields, there is a greater need than ever to secure this information while it is being transferred.Cryptography serves that purpose. The study of cryptographic procedures with the goal of identifying flaws, specifically the ability to decrypt encrypted messages without the need for a decryption key, is known as cryptanalysis, in contrast. Cryptosystems are made up of numerous intricate processes that guarantee data confidentiality, integrity, and authenticity in order for cryptography to resist cryptanalysis

Problem Statement

The existing cryptographic systems often face a trade-off between speed and security. Symmetric encryption techniques like AES are fast but may not be secure enough on their own, while asymmetric techniques like ECC offer strong security but are computationally intensive. This research aims to address these limitations by implementing a hybrid cryptosystem that leverages the strengths of both AES and ECC.

Challenges

Balancing speed and security is crucial in ensuring that the hybrid system provides both fast encryption/decryption processes and robust security. Integrating AES and ECC into a seamless cryptographic system presents implementation complexity, requiring careful design to leverage the strengths of both algorithms. Additionally, managing the secure exchange and storage of keys in the hybrid system is a key challenge. Performance evaluation involves measuring and comparing the efficiency and security of the hybrid system against other cryptographic methods to ensure its effectiveness.

Purpose and Objectives

The primary purpose of this research is to develop and implement a hybrid cryptosystem that combines AES and ECC to achieve both speed and security. The specific objectives are to design a hybrid cryptosystem by integrating AES for data encryption and ECC for secure key exchange, evaluate performance through comprehensive tests to measure the efficiency and security of the hybrid system, compare the hybrid system against other cryptographic methods in terms of speed and security, and enhance security measures by testing the system's resilience to various types of attacks.

Organization of Work for Each Chapter

Chapter 1: Generalities on Cryptography

This chapter provides a comprehensive introduction to the field of cryptography. It begins with a discussion of basic cryptographic vocabulary, including key terms such as cryptology, cryptanalysis, and encryption. The chapter then outlines the primary goals of cryptography: confidentiality, integrity, authentication, and non-repudiation. Following this, it delves into the types of cryptography, differentiating between classic and modern methods, as well as symmetric and asymmetric encryption. Through this foundational knowledge, readers are prepared to understand the more complex topics covered in later chapters.

Chapter 2: State of the Art on AES and ECC

In this chapter, we explore the two main cryptographic algorithms that form the basis of our hybrid system: Advanced Encryption Standard (AES) and Elliptic Curve Cryptography (ECC). The chapter begins with a detailed presentation of AES, covering its principles, operations, and advantages and disadvantages. It then transitions to an in-depth look at ECC, discussing its mathematical foundations, key generation, and applications. The chapter concludes with a comparison of the strengths and weaknesses of AES and ECC, setting the stage for the hybrid approach discussed in subsequent chapters.

Chapter 3: Our Contribution

This chapter details the novel hybrid cryptosystem developed in this research. It begins by explaining the methodology behind the integration of AES and ECC, highlighting the selection of algorithms and the design of the cryptosystem. The chapter includes practical examples to illustrate key generation, encryption, and decryption processes within the hybrid system. By providing a step-by-step guide to the implementation, this chapter demonstrates how the hybrid system combines the speed of AES with the security of ECC, offering a robust solution to modern cryptographic challenges.

Chapter 4: Implementation and Tests

In this chapter, we present the practical aspects of implementing the hybrid cryptosystem and the results of our performance tests. We start by describing the development environment and the tools used for coding and testing. The chapter then presents the results of various tests, including execution time for encryption and decryption, as well as key generation times for different data sizes. These results are compared with other cryptographic methods to highlight the efficiency of our hybrid system. Additionally, we discuss the system's security against common attacks, validating its robustness through extensive testing. The chapter concludes with a discussion on the practical implications of these findings and potential areas for further research.

Chapter 1

Generalities on Cryptography

introduction

Cryptography is the science and practice of securing communication and information through the use of codes and ciphers. It plays a crucial role in ensuring the confidentiality, integrity, and authenticity of data in various communication systems, including computer networks, the internet, and electronic transactions. The main goal of cryptography is to enable secure communication in the presence of malicious third parties, often referred to as adversaries.
In this chapter, we will see basic cryptography vocabulary and Goals of cryptography and Types of cryptography (classic and modern) and symmetrical and asymmetrical encryption.

1.1 Basic Cryptography Vocabulary

1. Cryptology: Is the science that studies secret communications. It consists of two related fields, cryptanalysis and cryptography.
2. Cryptanalysis: It is the art of revealing plain texts which have been the subject of encryption without knowing the key used to encrypt the plaintext.
3. Cryptography: This branch brings together all the methods which make it possible to encrypt and decrypt plaintext to make it incomprehensible to anyone not in possession of the key to use to decrypt it.
4. Plain text: the original readable text to be transformed to ciphertext.
5. Ciphertext: Also called cryptogram, it is the result of the encryption of the plain text.
6. Encryption: Is the method or algorithm used to transform text into plaintext in ciphertext.
7. Decryption: Is the method or algorithm used to transform text encrypted in plaintext.
8. Key: a secret string generated using the key generation algorithm.

1.2 Goals of cryptography

1.2.1 Confidentiality

The objective of confidentiality is to guarantee that solely authorized individuals or systems can access and comprehend the information being conveyed. In cryptographic terms, confidentiality is attained through a process known as encryption. Encryption converts plaintext into ciphertext, rendering it unintelligible without the corresponding decryption key(Adomey (2017)).

1.2.2 Integrity

Integrity guarantees that the data stays unaltered and has not undergone tampering during transmission or storage. Cryptographic methods, including cryptographic hash functions and message authentication codes, are utilized to accomplish this objective(Adomey (2017)).

1.2.3 Authentication

Authentication confirms the legitimacy of the sender or the source of the message. Cryptographic methods, such as digital signatures and message authentication codes, furnish authentication mechanisms to verify the identity of the sender(Adomey (2017)).

1.2.4 Non-repudiation

Non-repudiation prohibits the sender of a message from disavowing the authenticity of that message or any associated transaction. This objective is realized through the utilization of digital signatures and other cryptographic techniques, offering robust evidence of the message's origin and integrity(Adomey (2017)).

1.3 Types of cryptography

The types of cryptography are divided into two categories: classical and modern.

1.3.1 Classic cryptography

Cryptography has a rich history that dates back thousands of years. Ancient civilizations, such as the Egyptians and Greeks, used simple substitution ciphers to protect sensitive information. Over time, more advanced cryptographic techniques were developed, including the famous Caesar cipher and the invention of the Vigenère cipher during the Renaissance(Adomey (2017)).

The modern era of cryptography began during World War II with the development of complex encryption systems, such as the Enigma machine used by the Germans. After the war, cryptography evolved alongside advancements in technology, leading to the creation of public-key cryptography in the 1970s, a revolutionary concept that laid the foundation for secure digital communication(Adomey (2017)).

An ultimate objective of cryptology is authentication. It involves an individual proving that a message was genuinely sent by them. This concept closely aligns with identification, where, additionally, a message is involved. Authentication is predominantly handled in cryptography through public key signature schemes. The adoption of public key signature schemes, as opposed to hash functions or secret key methods, enables non-repudiation. This means that a valid signature for a given message cannot be disputed by the signatory(Banoth and Regar (2023)).

1) Substitution Cryptography

The substitution cipher involves systematically replacing one letter with another. An example of this is a function that shifts the letters of a sentence by n positions. This type of substitution is called monoalphabetic. Indeed, there is another type of substitution known as polyalphabetic substitution, which introduces the concept of keys(Martin (2004)).

- **Examples of substitution encryption:**

Caesar Cipher(101-44 BC) This encryption method is considered the oldest form of monoalphabetic substitution cipher. Its principle is quite elementary—it simply involves shifting the letters of a message three positions to the right in the Latin alphabet. Thus, the letter A is transformed into D, B into E, and so on(Ramdane and Mokrane (2008)).

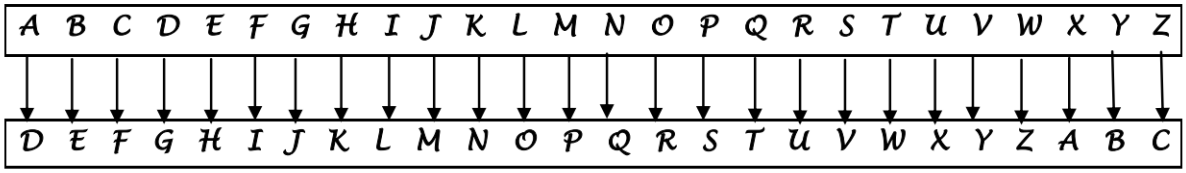


Figure 1.1: Caesar cipher: The letters of the plain text message are shifted by three positions in the alphabet.

Code of Vigénère (1523-1596) Polyalphabetic substitutions are combinations of several simple substitution tables, where the table is changed for each letter, which makes the message encrypted harder to break without the key.

One of the most well-known polyalphabetic substitution ciphers is the Vigenère Cipher. In the 16th century, Blaise de Vigenère introduced a novel encryption method, drawing significant inspiration from the works of the German abbot Jean Trithème of the same century. Although Vigenère's approach was influenced by Trithème, he made a slight modification by introducing the use of an encryption key, thereby enhancing the security of the method (Leary (1996)).

Certainly! Let's apply the Vigenère cipher to the clear message "HELLO" with the key "KEY":

Repeat the key to match the length of the clear message: Clear message: HELLO
Key: KEY
Convert the letters to numbers using a zero-based numbering system:

- 'H' (7) + 'K' (10) = 'R' (17)
- 'E' (4) + 'E' (4) = 'I' (8)
- 'L' (11) + 'Y' (24) = 'I' (35 mod 26 = 9)
- 'L' (11) + 'K' (10) = 'U' (21)
- 'O' (14) + 'E' (4) = 'S' (18)

So, using the Vigenère cipher with the key "KEY", the clear message "HELLO" encrypts to the cipher-text "RIEUS"(Leary (1996)).

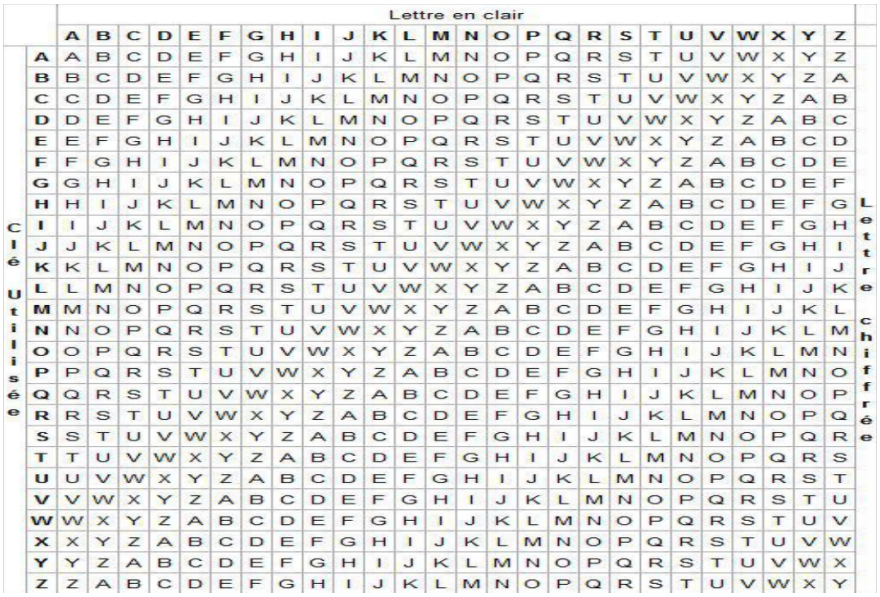


Figure 1.2: Vigenère table.

Transposition cryptography Transposition is a means of diffusion according to Shannon's principle. It enables the scattering of redundancies or plaintext by distributing them throughout the ciphertext.

The principle is then very simple: the letters are mixed up, their identity is preserved (an L remains an L), but they change position. Below is an example of encryption using a simple columnar transposition, which involves writing a text line by line in a 10-column table, then transmitting the message by column. Let's encrypt the following message:" POURQUOI REMETTRE A DEMAIN CE QUE ON PEUX FAIRE A UNE MAIN ".

The sender transmits the message by columns, resulting in the following sequence of characters:
"PMAPU OEIEN UTNUE TCXM QREFA UEQAI OAUIN IDER REOE EMNA"

1.3.2 Modern cryptography

Modern cryptography is divided into two parts according to the type of key (symmetric/asymmetric) and the type of data flow (stream / block).(Martin (2004))(Ramdane and Mokrane (2008))W.Stallings (2014).

1) Symmetric Cryptography

Symmetric cryptography uses the same secret key to encrypt and decrypt data, ensuring its confidentiality. Symmetric algorithms are very fast in terms of computation. However, they pose the challenge of key distribution between a sender and a receiver. Sharing a key with each communicating entity in a group of n entities is di icult and leads to a large number of keys to manage(Lebsir et al. (2020)).
Symmetric encryption schemes can be classified into two categories, the stream cipher and block cipher:

Stream Cipher Stream cipher, also known as stream encryption, operates on each element of the plaintext (character, bits) as they arrive. Its structure relies on a pseudo-random number generator whose output is combined via XOR with the information to be encrypted. These pseudo-random numbers are generated from a secret key(Dumont (2010)).

Block Cipher The basic idea of block cipher encryption is to divide the text into relatively large blocks, typically 64 or 128 bits, and encode each block separately. The same encryption key is used for each block, and it is the encryption key that determines the order in which substitution, permutation, and other mathematical functions are performed on each block(Alimzhanova et al. (2022)). There is many Types of Block Ciphers for examples: Substitution-Permutation Networks (e.g., AES),Feistel Ciphers (e.g., DES),Block Cipher Modes of Operation (e.g., CBC, ECB),...

Advantages:

1-Strong security due to the complexity of the encryption algorithm.
2-Well-suited for encrypting large blocks of data.

Disadvantages:

1-Higher latency compared to stream ciphers.
2-Requires padding for data that does not fit the block size.

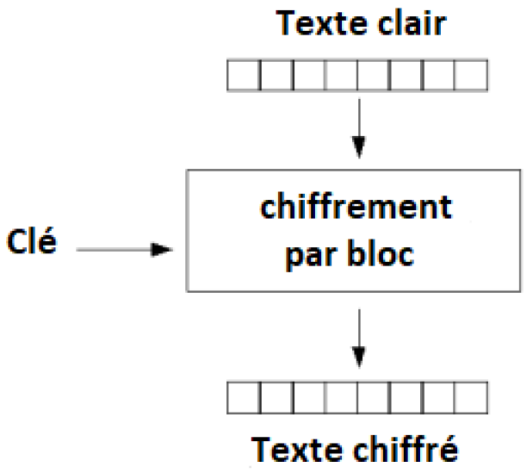


Figure 1.3: Diagram illustrates Block encryption.

- **Modern Symmetric Algorithms**

Symmetric cryptography, also known as secret-key cryptography, involves the use of the same key for both encryption and decryption of data. This key must be shared between the communicating parties and kept secret to ensure the security of the information.

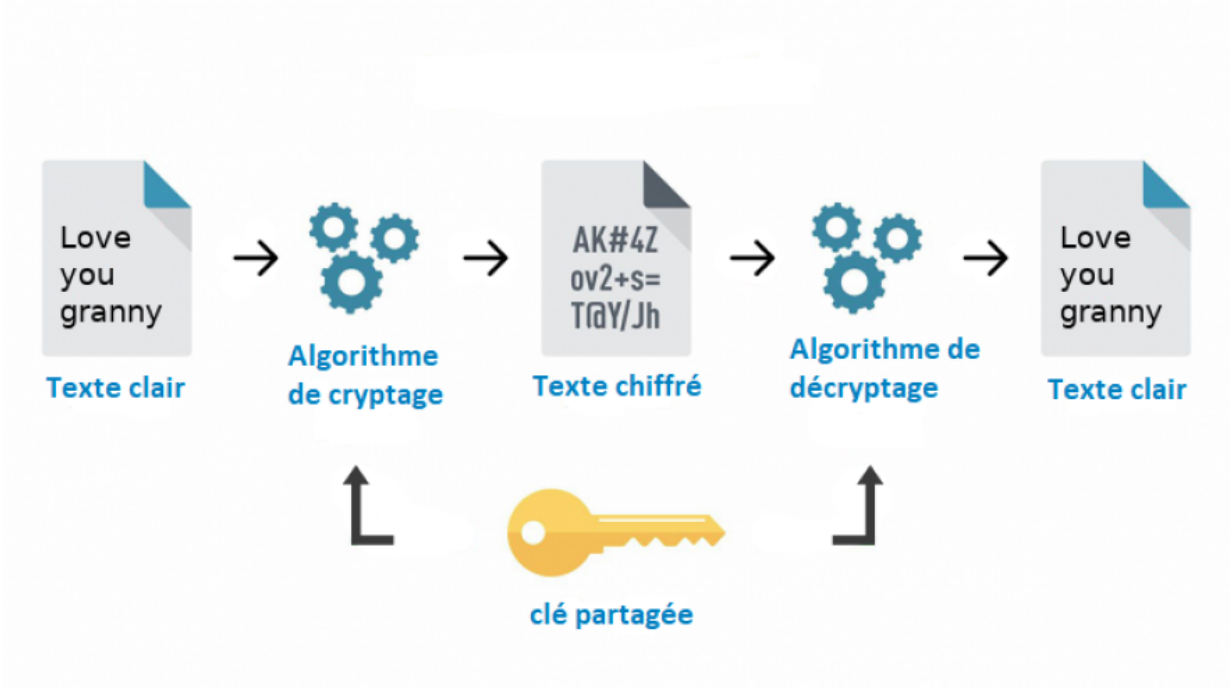


Figure 1.4: Diagram illustrates symmetric encryption.

DES: Data Encryption Standard Until the introduction of the Advanced Encryption Standard (AES) in 2001, the Data Encryption Standard (DES) was the most widely used encryption scheme. It was designed

in the 1970s by the American firm IBM to be a block cipher encryption system with a secret key and was adopted as an encryption standard in 1977.

As illustrated in Figure 1-4, DES begins with an initial bit permutation (IP) using the permutation matrix illustrated , performs Feistel ciphering using sub-keys generated by a key generation program, and finally executes the inverse initial permutation (IP^{-1}) using a permutation matrix.

DES processes messages of 64 bits with keys of 56 bits, and it produces a ciphertext of 64 bits as output. It achieves this by iterating the Feistel structure 16 times (Stallings (2006a)).

AES: Advanced Encryption Standard AES is the most widely used cipher in the world. This symmetric cipher was standardized by NIST in 2000, replacing DES. Nowadays, almost all modern processors integrate AES instructions, including Intel processors, AMD processors, rather than many other software products such as OpenSSL (Aumasson (2017)).

Contrary to the majority of asymmetric encryption algorithms, whose security relies on challenging mathematical problems such as the discrete logarithm in the case of ECC or integer factorization in the case of RSA, AES derives its strength from the combination of permutation and substitution, more commonly known as substitution-permutation networks (SPN). We can say that AES itself poses a difficult problem, as many qualified individuals have attempted to break AES encryption and failed (Stinson (2005)).

AES is a symmetric block cipher that takes a plaintext block size of 128 bits or 16 bytes. The key length can be 128, 192, or 256 bits (Stallings (2006b)).

- **Advantages and Disadvantages of Symmetric Cryptography**

Advantages	**Disadvantages**
Fast Encryption/Decryption	Secret key exchange problem
Large volumes of data to encrypt	$n(n-1)/2$ keys for n partners
Relatively short keys	No electronic signature

Table 1.1: Advantages and Disadvantages of Symmetric Encryption

2) Asymmetric Cryptography

Asymmetric cryptography, also known as public-key cryptography, relies on the use of a public key (which is distributed) and a private key (kept secret), with one key used for encryption and the other for decryption. Thus, the sender can utilize the recipient's public key to encrypt a message that only the recipient (in possession of the private key) can decrypt, ensuring the confidentiality of the c ontent. Conversely, the sender can use their own private key to encrypt a message, and the recipient can decrypt it with the public key; this is the mechanism used by digital signatures to authenticate the author of a message. In asymmetric cryptography, it is impossible to derive the private key from the public key(Nawal (2017)).

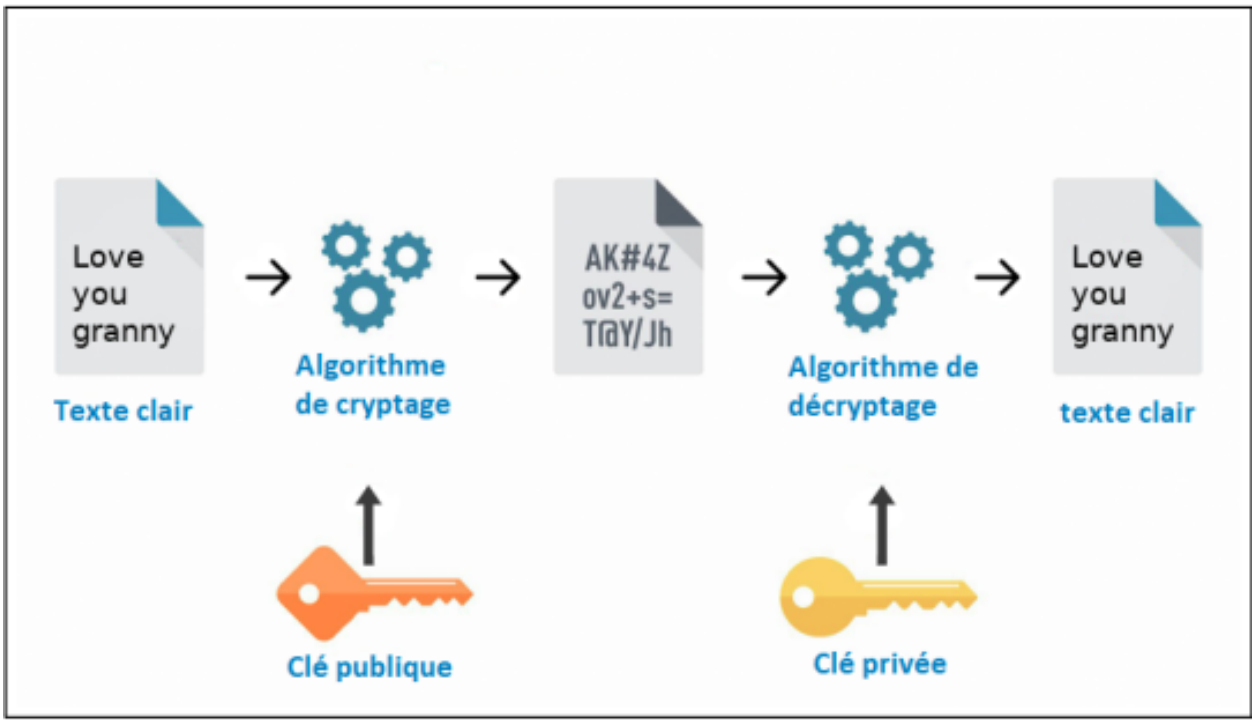


Figure 1.5: Diagram illustrates asymmetric encryption.

- **Modern Asymmetric Algorithms**

RSA: Rivest Shamir Adleman A groundbreaking encryption method is introduced, characterized by the unique property that disclosing an encryption key publicly does not divulge the corresponding decryption key. This brings about two significant outcomes:

1. There is no need for couriers or other secure channels to transmit keys, as a message can be encoded using an encryption key made public by the intended recipient. Only the recipient can decode the message, as they alone possess the corresponding decryption key.

2. Messages can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly disclosed encryption key. Signatures are immune to forgery, and a signer cannot later disown the validity of their signature.

This innovation finds immediate applications in electronic mail and electronic funds transfer systems. To encrypt a message, it is represented as a number M, raised to a power e as specified publicly, and then the remainder is taken when the result is divided by the publicly specified product, n, of two large secret prime numbers, p and q.

Decryption follows a similar process, utilizing a different secret power d, where $e{\cdot}d \equiv 1 \pmod{(p-1)(q-1)}$. The security of the system hinges, in part, on the complexity of factoring the published divisor, n (Rivest et al. (1978)).

ECC: Elliptic Curve Cryptography To construct a cryptographic system using elliptic curves, we need to identify a "hard problem" analogous to the factorization of the product of two prime numbers as in the "RSA" method or the discrete logarithm problem. Consider the equation $Q = kP$ where $Q, P \in E(a, b)$ and $k > p$. It is relatively easy to compute Q given k and P, but it is relatively difficult to determine k given Q and P. This is known as the discrete logarithm problem for elliptic curves.

We give an example, consider the group $E_{23}(9, 17)$, which is the group defined by the equation $y^2 \mod 23 = (x^3 + 9x + 17) \mod 23$. What is the discrete logarithm k of $Q = (4, 5)$ with respect to the base $P = (16, 5)$ The brute-force method involves calculating multiples of P until Q is found.

$$\begin{aligned} P &= (16, 5); \\ 2P &= (20, 20); \\ 3P &= (14, 14); \\ 4P &= (19, 20); \\ 5P &= (13, 10); \\ 6P &= (7, 3); \\ 7P &= (8, 7); \\ 8P &= (12, 17); \\ 9P &= (4, 5) \end{aligned}$$

Because $9P = (4, 5) = Q$, the discrete logarithm of $Q = (4, 5)$ with respect to the base $P = (16, 5)$ is $K = 9$. In a real-world application, K would be so large that the brute-force approach would be infeasible(Hankerson and Menezes (2021)).

- **Advantages and Disadvantages of Asymmetric Cryptography**

Advantages	**Disadvantages**
Authentication of messages by electronic signature	Slow to run due to load calculation
No need to share secret keys via a secure transmission channel	Possible key substitution attacks
n key pairs for n partners	The key length greatly increased

Table 1.2: Advantages and Disadvantages of Asymmetric Encryption.

1.3.3 Hybrid Cryptography

Hybrid cryptography is a cryptographic system that combines both major families of cryptographic systems: symmetric cryptography and asymmetric cryptography.

A hybrid cryptosystem leverages the advantages of both symmetric and asymmetric encryption, such as the speed of symmetric cryptography and the ability to transmit the secret key through asymmetric cryptography.

The principle is simple: key exchange for symmetric encryption is facilitated through public-key cryptography, and the data to be exchanged is encrypted using a symmetric encryption algorithm, resulting in efficient communication.

1) The most famous hybrid cryptosystems:

- **GnuPG (GNU Privacy Guard)**

GnuPG is a hybrid encryption software because it employs a combination of symmetric-key cryptography for encryption speed and public-key cryptography to facilitate secure key exchange, typically by using the

recipient's public key to encrypt a session key that is used only once. This mode of operation is part of the OpenPGP standard.
GnuPG supports the following algorithms:
Asymmetric algorithms: RSA, ElGamal, DSA.
Symmetric algorithms: DES, IDEA, CAST5, Blowfish, AES, Camellia(Nguyen (2004)).

- **PGP (Pretty Good Privacy)**

PGP is cryptographic encryption software, developed and distributed in the United States by Philip Zimmerman in 1991.
PGP aims to provide confidentiality and authentication for data communication. It is commonly used for data signing, encryption, and decryption of texts, emails, and files. Utilizing both asymmetric and symmetric cryptography, it is classified as hybrid cryptography software.

In the operation of PGP, a secret key (known as a session key, valid for a single file or message) is generated. The message or file is encrypted using this session key with a symmetric cryptography algorithm. Then, this secret key is encrypted using the recipient's RSA or ElGamal public key and added to the beginning of the message or file. The recipient of the message decrypts the message header with their RSA or ElGamal private key and extracts the secret key, allowing them to decrypt the message.

PGP offers a selection of multiple secret key algorithms for encrypting the message. By secret key algorithm, we refer to symmetric encryption using the same key for encryption and decryption. The three symmetric block ciphers offered by PGP are CAST, triple DES, and IDEA, all of which operate on plaintext and ciphertext blocks of 64 bits. The key sizes for CAST and IDEA reach 128 bits, and for triple DES, it is 168 bits(Zimmermann (1997)).

1.4 Conclusion

Given the necessity of security in multimedia, we devote an entire chapter to it, focusing on encryption algorithms and their operation. Both basic cryptosystems, secret key and public key, have their own problems and peculiarities. The strength of public key algorithms lies in key distribution, while secret key algorithms excel in encryption speed. Therefore, increasing the speed and security of cryptosystems undoubtedly involves the combined use of these two techniques, known as hybrid cryptography.

In the next chapter, we will detail the strongest algorithm in symmetric encryption, AES , as well as elliptic curve-based encryption.

Chapter 2

State of the art on AES and ECC

Introduction

Modern cryptography is made up of two major families of cryptography, each of them has positive points and negative points, symmetric cryptography or secret key cryptography is characterized by the speed of a symmetric cryptosystem which uses few system resources, asymmetric cryptography or key cryptography public allows the transmission of the secret key.
In this chapter, we will talk about the strongest algorithm in the family of symmetric cryptography which is AES which provides a level satisfactory security and rapid encryption in terms of execution time, and cryptography based on elliptic curves which is based on the discrete logarithm problem and that it represents the strongest cryptosystem in the family of cryptography asymmetric.

2.1 AES (Advanced Encryption Standard)

In this section we will present and detail the operating principle of the AES.

2.1.1 Presentation of the AES

Since November 26, 2001, the "Rijndael" block encryption algorithm, in its 128-bit version, became the successor to DES under the name Advanced Encryption Standard (AES)(Dubois (2017)).

From a competition launched by the National Institute of Standards and Technology (NIST) in 1997, Rijndael passed all the selection stages and now has a federal standard American registered under FIPS number 197. Registered with the National Security Agency (NSA)(Dubois (2017)).

Promoted by the US government, to become a standard for secure exchange information, within the United States and between the United States and its partners. Field application of the AES has evolved and becomes, from October 1, 2015, the algorithm of information encryption up to the TOP SECRET level in the United States. Likewise, he is, today, the symmetric block encryption algorithm most commonly used in the West. AES is a symmetric block encryption algorithm. It encrypts and decrypts blocks of data using a single key.

Unlike DES, based on a Feistel network, AES relies on a network of substitutions and permutations (SP-network). The latter is made up of functions of nonlinear substitutions contained in an S-Box and linear permutation functions. Each box takes a block of text and the key as input then provides a block of ciphertext in exit.

The AES inputs and outputs are 128-bit blocks and the key length can be be 128, 192 or 256 bits.

For its internal operation, each data block is organized into an array of four columns and four rows, each box containing one byte, i.e. $4 \times 4 \times 8 = 128$ bits per table. Encryption and decryption operations are performed on this table, then, the result is copied into an output table(Dubois (2017)).

2.1.2 Principle of AES

AES is a symmetric block cipher that takes a plaintext block size of 128 bits or 16 bytes. The key length can be 128, 192, or 256 bits. The input for both encryption and decryption algorithms is a single 128-bit block. This block is represented as a 4x4 square matrix of bytes. Similarly, the key is represented as a square matrix of bytes, and then the key expansion function generates N+1 round keys. Each round key serves as one of the inputs to the AddRoundKey transformation in each round (W.Stallings (2014)).

The first N-1 rounds consist of four distinct transformation functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey, which are described later. The final round contains only three transformations, and there's a single initial transformation (AddRoundKey) before the first round, which can be considered as round 0. The output of the final round is the encrypted 128-bit text(W.Stallings (2014)).

The encryption consists of N rounds, where the number of rounds depends on the key length: 10 rounds for a 128-bit key, 12 rounds for a 192-bit key, and 14 rounds for a 256-bit key.

The pseudo-code for the AES encryption function can be written as follows. In this case, Nb corresponds to the number of 32-bit words and consequently to the number of columns in the state array, and Nr corresponds to the number of rounds used in the algorithm(W.Stallings (2014)).

Algorithm 1 Cipher function

```
function CIPHER(byte in[4 × Nb], byte out[4 × Nb], word w[Nb × (Nr + 1)])
    byte state[4][Nb]
    state ← in
    AddRoundKey(state, w[0, Nb − 1])
    for round = 1 to Nr − 1 do
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round × Nb, (round + 1) × Nb − 1])
    end for
    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr × Nb, (Nr + 1) × Nb − 1])
    return state
end function
```

2.1.3 AES transform functions

We'll now delve into each of the four transformations employed in AES. The encryption process relies on four fundamental operations: AddRoundKey, SubBytes, ShiftRows, and MixColumns. These operations are conducted on the state array. The encryption cycle encompasses an initial transformation, intermediate rounds, and a final round. For decryption, the inverse operations of the four encryption functions are applied in reverse order: InvShiftRows, InvSubBytes, and InvMixColumns. However, the AddRoundKey function remains unaltered(Li et al. (2009)).

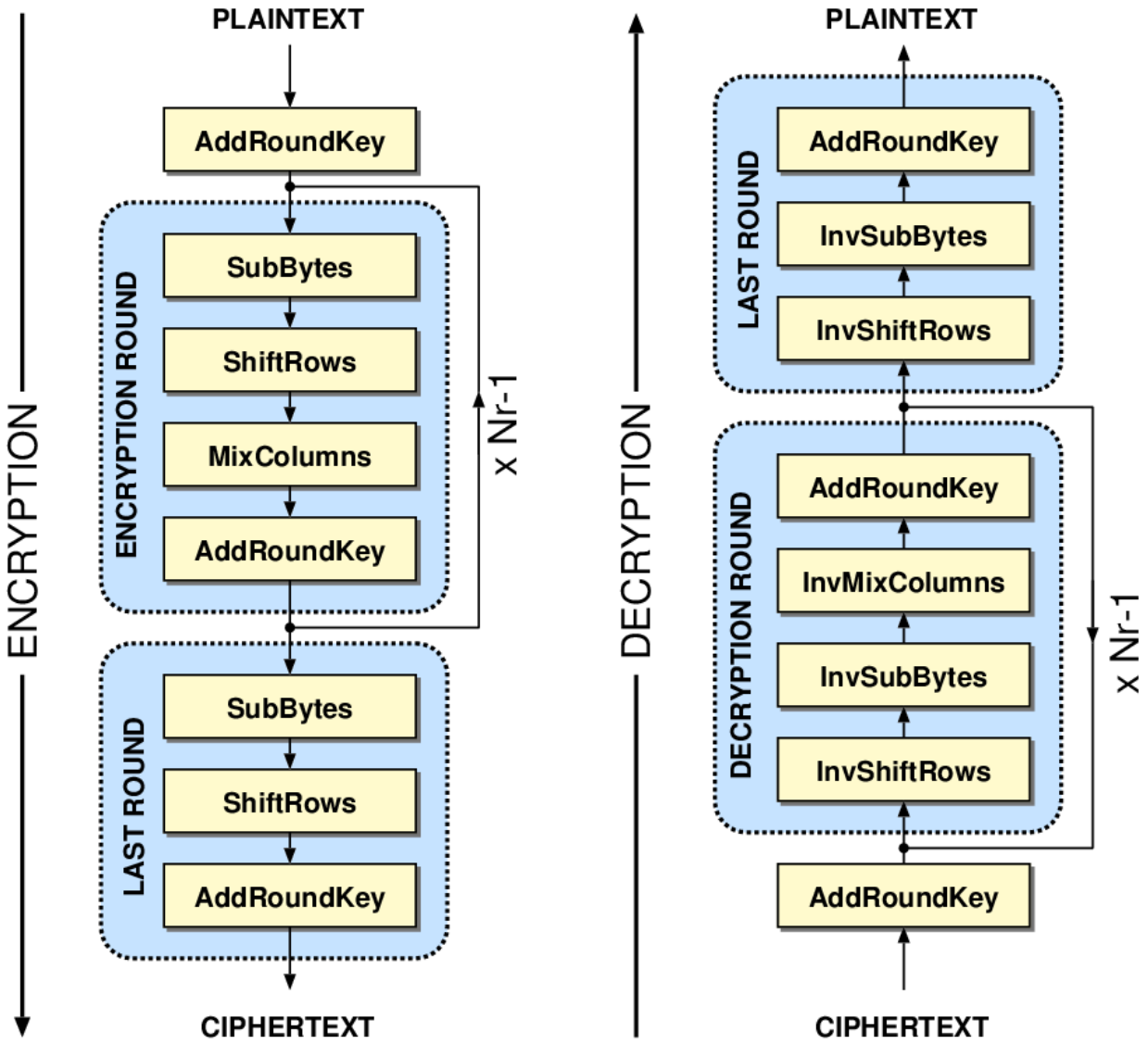


Figure 2.1: AES Encryption and Decryption. (W.Stallings (2014))

1) SubBytes/InvSubBytes

The byte substitution transformation, formerly known as SubByte, is a straightforward table lookup process. AES (Advanced Encryption Standard) defines a 16 * 16 matrix of byte values, referred to as an S-box, which encompasses a permutation of all 256 possible 8-bit values. Each individual state byte is remapped to a new byte in the following manner: the 4 leftmost bits of the byte serve as the row value, while the 4 rightmost bits serve as the column value. These row and column values act as indices within the S-box, selecting a unique 8-bit output value. For instance, the hexadecimal value [95] corresponds to row 9, column 5. In the decryption process, known as InvSubByte, the same steps as SubByte are followed, except that the matrix employed is the Inverse S-box, facilitating the required permutations(Li et al. (2009))

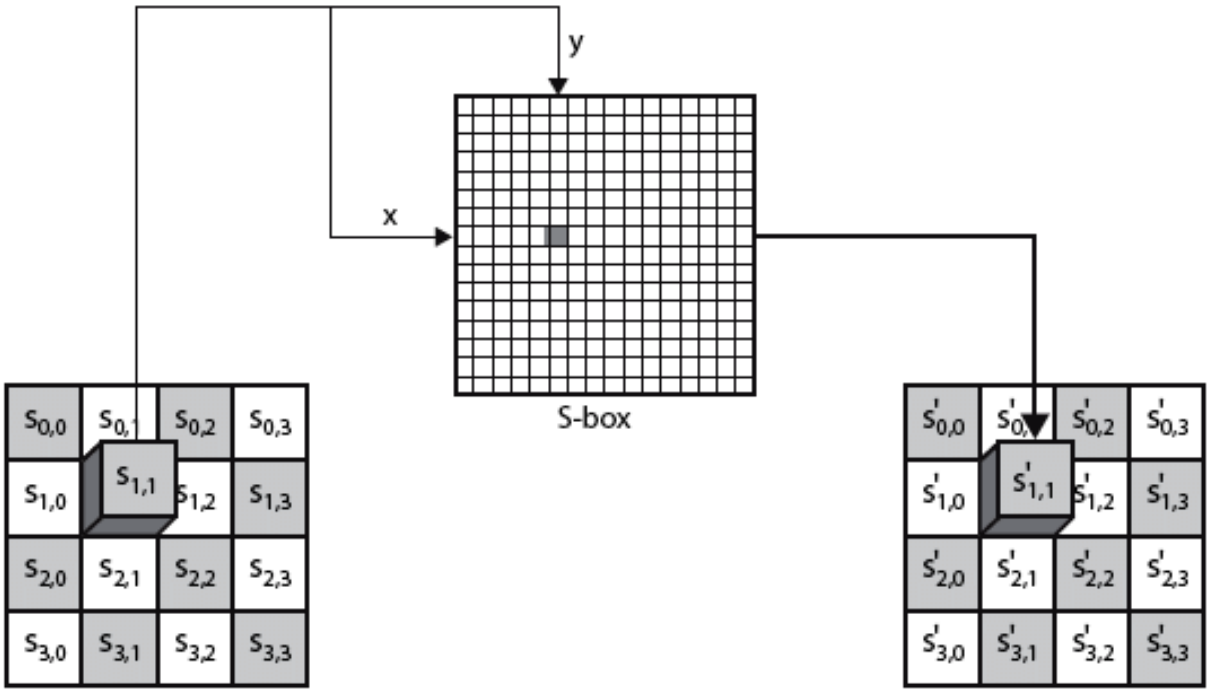


Figure 2.2: SubBytes/InvSubBytes transformation(W.Stallings (2014))

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 2.3: S-Box transformation table (W.Stallings (2014))

2) ShiftRows/InvShiftRows

The transformation of the front shift line, called ShiftRows, is illustrated in Figure 14. The first state line remains unchanged. For the second line, a circular left shift of 1 byte is performed. For the third line, a circular left shift of 2 bytes is performed. For the fourth line, a circular left shift of 3 bytes is performed.

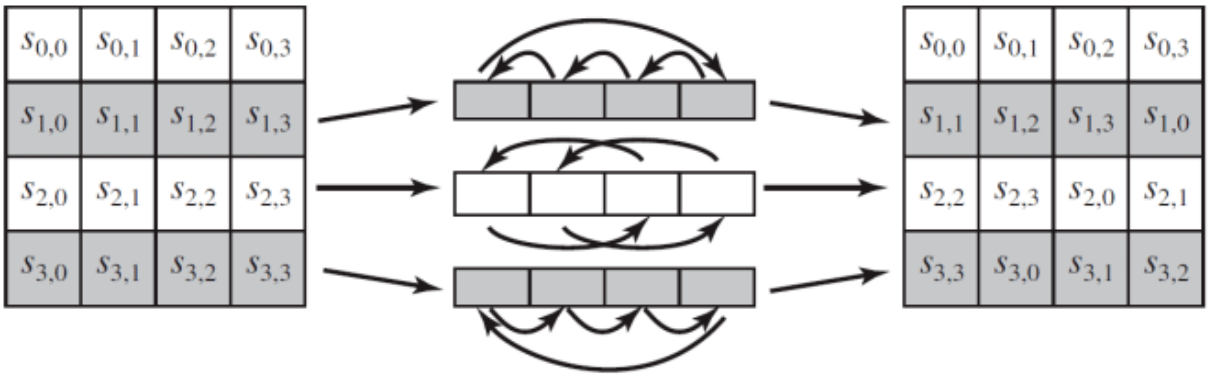


Figure 2.4: ShiftRows transformation. (W.Stallings (2014))

The inverse shift row transformation, called InvShiftRow, performs circular shifts in the opposite direction for each of the last three rows, with a circular shift of 1 byte to the right for the second row, and so on(Al-Wattar et al. (2015))

3) MixColumns/InvMixColumns

The MixColumns transformation, also known as direct mixing column, operates on each column individually. Each byte of a column is mapped to a new value that is a function of the four bytes of that column. The transformation can be defined by multiplying a specific matrix with the state matrix. Each element of the product matrix is the sum of the products of the elements from a row and a column.

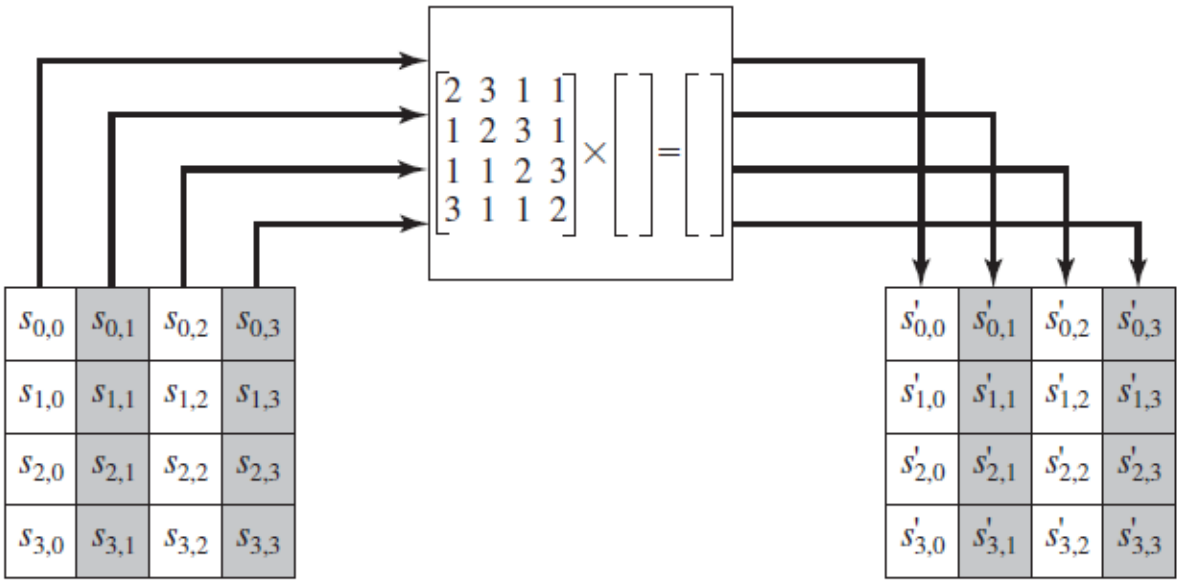


Figure 2.5: MixColumns Transformation (W.Stallings (2014))

The reverse mixing column transformation, called InvMixColumn, is defined by the following matrix multiplication:

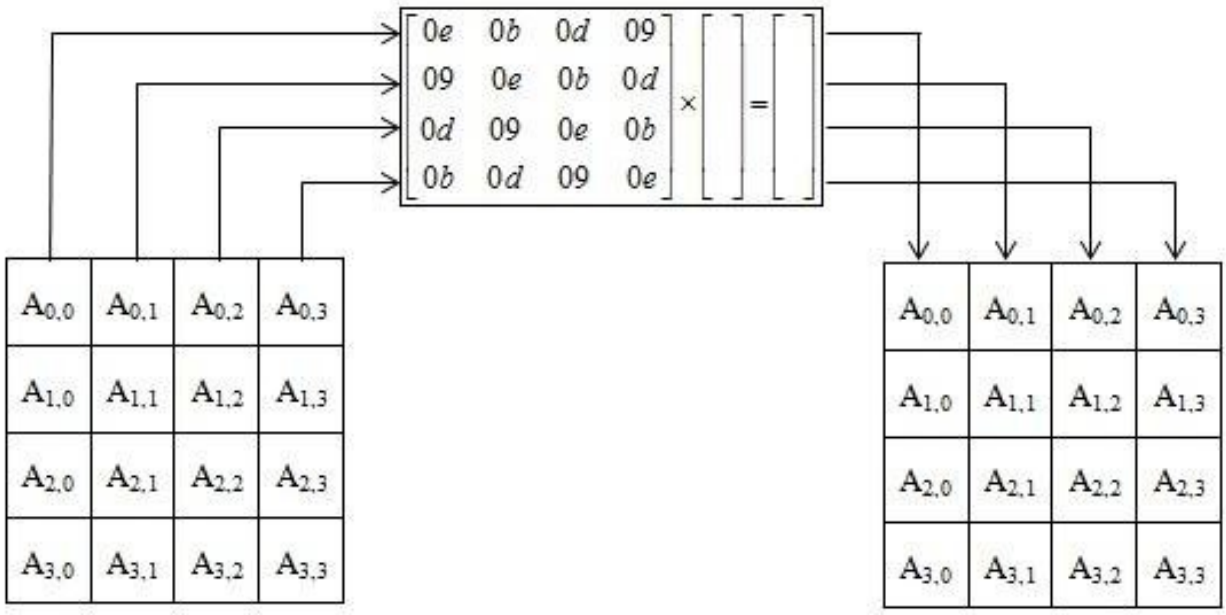


Figure 2.6: InvMixColumns Transformation (W.Stallings (2014))

4) AddRoundKey

To perform the AddRoundKey transformation, a round key is added, and the 128 state bits are XOR bit-wise with the 128 bits of the round key. The operation is considered as a column operation between the 4 bytes of a state column and a word of the round key.

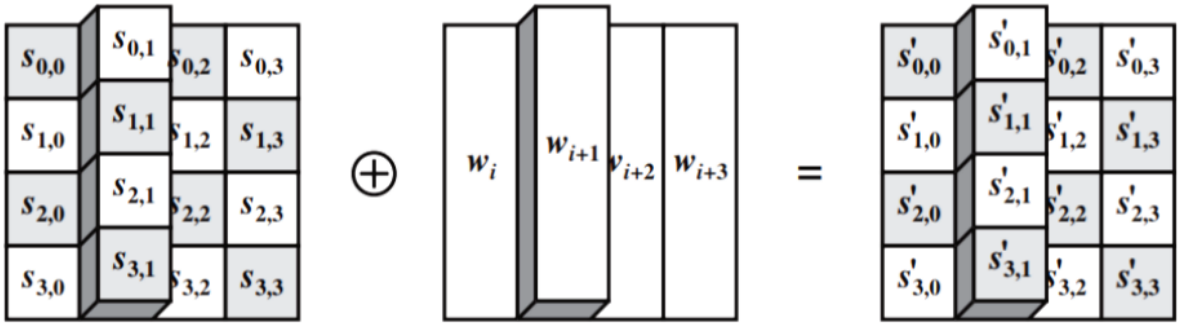


Figure 2.7: Round Key transformation(W.Stallings (2014))

5) Key expansion

As a reminder, in the AES algorithm,Nb = 4 words and Nr = (10, 12, 14) words with 1 word = 4 bytes = 32 bits.

The AES algorithm, using an encryption key K of 128, 192, or 256 bits, executes a key expansion procedure to generate round keys. The key expansion procedure generates a total of Nb(Nr+1) words. Indeed, the algorithm requires an initial set of Nb key words, and each of the Nr rounds also requires Nb key words to encrypt the data. The algorithm involves two functions: SubWord and RotWord, as well as a round constant Rcon.

The result of the key expansion procedure is an array containing 4-byte words called $[w_i]$ with $0 \leq i \leq Nb(Nr+1)$.

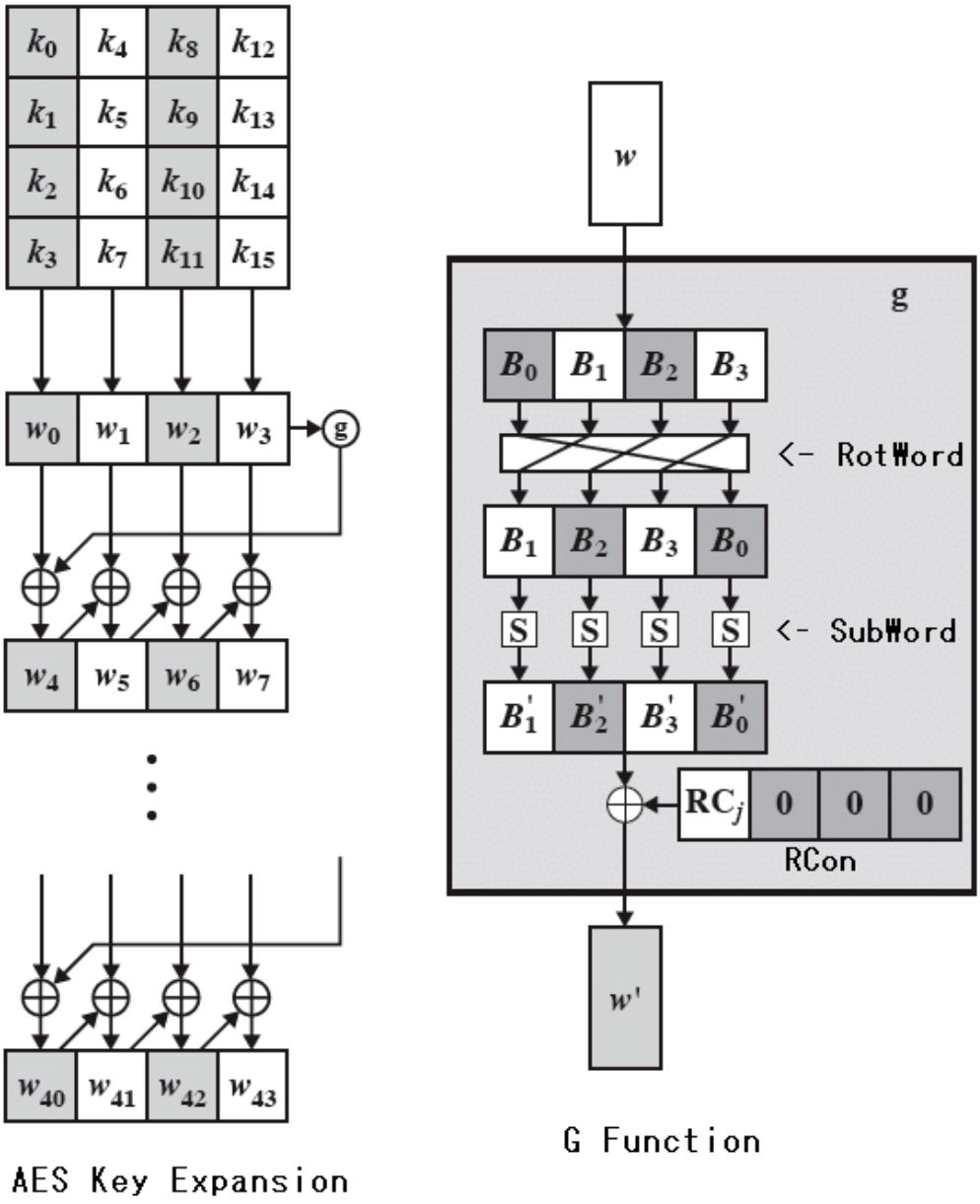


Figure 2.8: Overall key expansion algorithm(W.Stallings (2014))

2.1.4 Advantages and Disadvantages of AES

In this section we will present the advantages and disadvantages of AES, as well as solutions to eliminate these disadvantages:

1) Advantages

1. Fast encryption in terms of execution time.
2. Provides a very good level of security.
3. Its memory resource requirements are also very low.
4. Uses few system resources.

2) The inconvenient

1. Secret key exchange problem.
2. Key distribution problem n (n-1)/2 Keys for n partners.
3. Static tables: predefined SBOX INVSBOX RCON .

2.1.5 AES Encryption/Decryption With CBC

In this final phase, we implement the AES encryption and decryption algorithms utilizing Cipher Block Chaining (CBC) mode with a 256-bit key size to ensure robust security for our data. Before we delve into the encryption and decryption processes, let's first understand the basics of CBC mode.Vaidehi and Rabi (2014)

1) Cipher Block Chaining (CBC) Mode

CBC mode is a block cipher mode of operation for cryptographic protocols. In CBC mode, each plaintext block is XORed with the previous ciphertext block before encryption. This ensures that identical plaintext blocks do not result in identical ciphertext blocks, thus adding an extra layer of security.

1. **Initialization Vector (IV)**: CBC mode requires the use of an initialization vector (IV), which is a random value of the same block size as the cipher. The IV is XORed with the first plaintext block before encryption.
2. **Block Dependency**: CBC mode creates a dependency between blocks, meaning each ciphertext block depends on the previous ciphertext block. This dependency ensures that even small changes in the plaintext result in significant changes in the ciphertext.

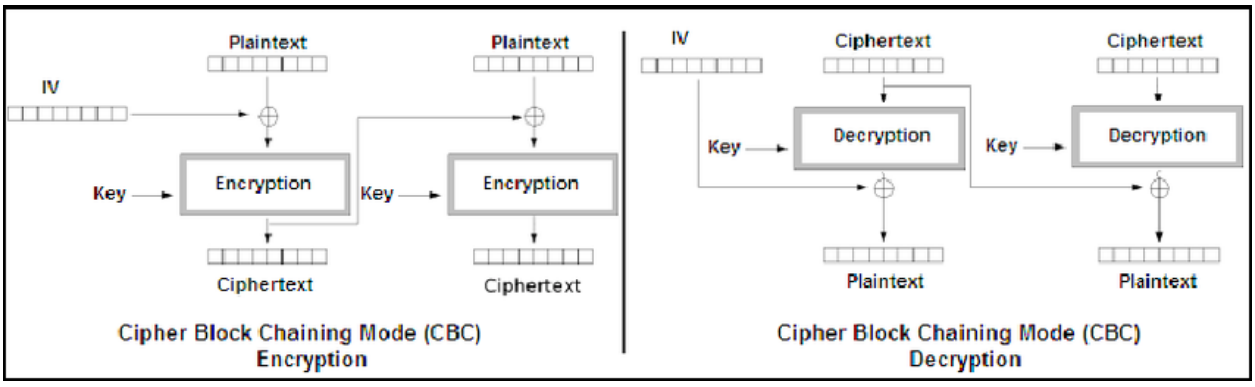


Figure 2.9: CBC-mode encryption and decryptio.

Now, let's proceed to outline the steps involved in both encryption and decryption processes using AES in CBC mode with a 256-bit key size.

2) Encryption Process

1. **Initialization Vector (IV) Generation**: A random IV is generated before encryption begins.
2. **Padding**: If the plaintext data length is not a multiple of the block size (256 bits for AES), padding is added.
3. **Dividing Plaintext into Blocks**: The plaintext data is divided into blocks.
4. **CBC Mode Encryption**: Each plaintext block is XOR with the previous ciphertext block before encryption.
5. **Final Ciphertext**: The final ciphertext is generated.

3) Decryption Process

1. **Initialization Vector (IV) Retrieval**: The IV used during encryption must be retrieved for decryption.
2. **CBC Mode Decryption**: Each ciphertext block is decrypted and XOR with the previous ciphertext block.
3. **Removing Padding**: If padding was added during encryption, it is removed.
4. **Final Plaintext**: The final decrypted plaintext is obtained.

4) Security Considerations

- **IV Reuse**: Never reuse the same IV with the same key.
- **Key Management**: Follow proper key management practices.
- **Cryptographic Strength**: AES with CBC mode and a 256-bit key size is considered secure when implemented correctly.

2.2 Elliptic Curve Cryptography (ECC)

Most products and standards that utilize public-key cryptography for encryption and digital signatures use RSA. Secure usage of RSA requires a considerable key length, which has burdened the processing overhead of applications utilizing RSA, with ramifications especially for e-commerce sites conducting significant secure transactions. A contender challenging RSA is elliptic curve cryptography (ECC). ECC emerges in standardization efforts, including the IEEE P1363 standard for public-key cryptography. ECC's main allure, compared to RSA, is that it appears to offer equivalent security for a much smaller key size, thus reducing processing overhead.W.Stallings (2014)

2.2.1 Using cryptography based on elliptic curves

1) Effectiveness of the elliptic curve

Elliptic curves provide good candidates for groups (Bos et al. (2016)):

- They have efficient arithmetic.
- There is no known attack better than generic attacks.
- Keys are easier to generate and smaller compared to RSA.

2) Equivalence between RSA and ECC

Existing work shows the equivalence of key sizes offering the same level of security. The table below compares the size of keys used in cryptography in RSA and ECC.

RSA	**ECC**
1024	160
2048	224
3072	256

Table 2.1: Key size comparison between RSA and ECC

We notice that elliptic curve cryptography allows the use of medium-sized keys compared to RSA while providing the same level of performance (Shou (2014)).

3) Applications that use ECC

Nowadays, several protocols and platforms utilize elliptic curve cryptography (ECC) due to its speed and security. Among these:

- **TLS:** The internet exchange security protocol uses ECC in the "Handshake" phase for agreeing on the key to use for each session.
- **Smart Cards:** Alongside RSA, new smart cards support ECC.
- **Biometric Passport:** ECC is used to construct the session key.
- **Bitcoin:** ECC is used for electronic Transport Layer Security (TLS) signature during Bitcoin payment transactions. The curve used by Bitcoin is Secp256k1 with the equation $y^2 = x^3 + 7$(Lotfi (2017)).

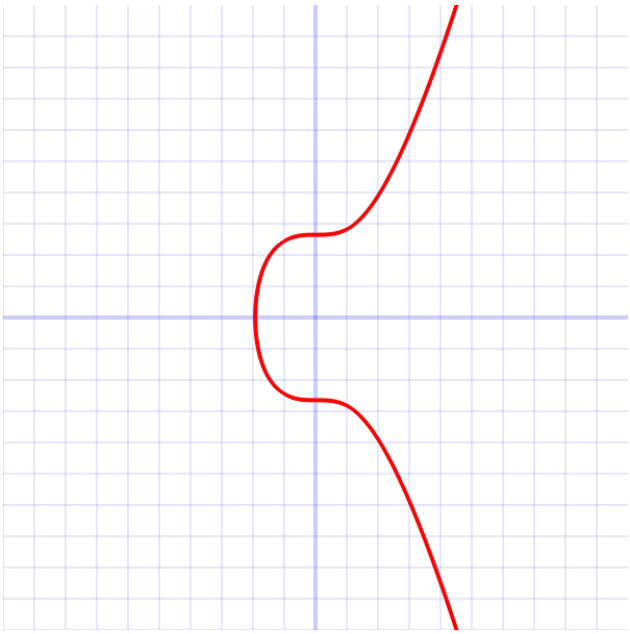


Figure 2.10: Secp256k1 curve used by Bitcoin.

2.2.2 General

This section provides an overview of elliptic curves and ECC. We begin with a brief review of the concept of an abelian group and cyclic group, then we examine the concept of prime and binary finite fields, followed by a look at elliptic curves defined over finite fields. Finally, we can explore elliptic curve cryptography.

1) Band

In mathematics, a group is a pair (E, $\cdot$) where E is a set and $\cdot$ is an internal composition law that combines two elements a and b from E to obtain a third element a $\cdot$ b. The law must satisfy the four axioms below(Shou (2014)).

1. Closure: $\forall$ (a, b) $\in$ E :a $\cdot$ b $\in$ E
2. Associativity : $\forall$(a, b) $\in$ E :(a $\cdot$ b) $\cdot$ c = a $\cdot$ (b $\cdot$ c)
3. Neutral element : $\exists$ e $\in$ E :a $\cdot$ e = e $\cdot$ a = a
4. Symmetrical : : $\forall$ a $\in$ E, $\exists$ b $\in$ E :a $\cdot$ b = b $\cdot$ a = e

- **Abelian group**

An Abelian group is a group in which the internal composition law is commutative. A set E is a commutative group when(Shou (2014)).
$\forall$ (a, b) $\in$E : a $\cdot$ b = b $\cdot$ a

- **Groupe cyclique**

A finite group G is cyclic if every element of the group can be expressed as a power or a multiple of a particular element g, called the generator of the group, that is,

$$G = \langle g \rangle = \{g^n \mid n \in \mathbb{Z}^*\}.$$

For example, if $G = \{g^0, g^1, g^2, g^3, g^4, g^5\}$ and $g^6 = g^0$, then G is a cyclic group. Every cyclic group is abelian because

$$g^n \cdot g^m = g^{n+m} = g^{m+n} = g^m \cdot g^n.$$

The order of an element e in a cyclic group is the smallest positive integer n such that $ne = 0$ (in additive notation) or $e^n = 1$ (in multiplicative notation). Let's take the same group G from the previous paragraph; for example, the order of the element g^2 is 3 because the neutral element of the group is $g^0 = 1$ and $(g^2)^3 = g^6 = 1$(Shou (2014)).

2) Body

A field is a set E equipped with two composition laws, denoted respectively by $+$ and $\cdot$. These laws must satisfy the following conditions:

- The pair $(E, +)$ forms an abelian group, where there exists a neutral element, denoted by 0, such that $\forall a \in E : a + 0 = 0 + a = a$.
- The pair $(E \setminus \{0\}, \cdot)$ also forms an abelian group with the neutral element being 1: $\forall a \in E : a \cdot 1 = 1 \cdot a = a$.
- The multiplication $\cdot$ distributes over addition, meaning:

$$\forall (a, b, c) \in E : a \cdot (b + c) = a \cdot b + a \cdot c \text{ and } (b + c) \cdot a = b \cdot a + c \cdot a.$$

In other words, a field is a ring whose non-zero elements form an abelian group under multiplication(Shou (2014)).

:
A finite field F is a field whose number of elements is fini te. The number of elements is the order of the field, denoted by q, which can be represented by the power of a prime number $q = p^n$, where p is a prime number, called the characteristic of the field, and $n \in \mathbb{Z}^+$. To study cryptography on elliptic curves, it is necessary for us to understand the two types of fields below(Shou (2014)).

- **Prime body**

:
A field is a prime field, denoted F_p, when the order of the field $q = p^n$ and p is a prime number. The field consists of the integers $\{0, 1, 2, ..., p-1\}$, and for all $a \in \mathbb{Z}$, $a \bmod p$ gives the unique remainder r that lies between $[0, p-1]$(Shou (2014)).

- **Binary body**

:
A finite field of order $2n$ is a binary field, denoted F_{2^n}, which can be constructed using a polynomial representation. The elements of the field are binary polynomials with coefficients $a_i \in \{0, 1\}$ and degrees less than n. That is:

$$F_{2^n} = \{a_{n-1}z^{n-1} + a_{n-2}z^{n-2} + \ldots + a_1 z + a_0 : a_i \in \{0, 1\}\}$$

2.2.3 Overview of elliptic curves

After presenting the necessary mathematical concepts, we will proceed, in this section, to define elliptic curves along with the set of operations that can be performed on them.

1) Weierstrass equation

A elliptic curve over K is defined as the set of solutions of the following Weierstrass equation:

$$E : Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3$$

Where the coefficients a_1, a_2, a_3, a_4, and a_6 are in K.

To simplify notions, we write the Weierstrass equation with non-homogeneous coordinates: $X = x/z$ and $Y = y/z$.

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6.$$

2) Definition of elliptic curves

Let K be a finite field. An elliptic curve over K is a curve in the projective plane, cubic and without singular points, equipped with a distinguished point which plays a particular role: the neutral element. It is defined by an irreducible homogeneous polynomial in three variables with coefficients in K. By a projective change of variables, we can always reduce it to an equation known as the Weierstrass form:(Martinez et al. (2005)) The equation of an elliptic curve in Weierstrass form is given by:

$$Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

where the coefficients a_1, a_2, a_3, a_4, and a_6 are in the field K and $\Delta \neq 0$ with:

$$\Delta = -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6$$

where:

$$d_2 = a_1^2 + 4a_2,$$
$$d_4 = 2a_4 + a_1 + a_3,$$
$$d_6 = a_3^2 + 4a_6,$$
$$d_8 = a_1^2 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2.$$

- **Finished bodies**

The condition $\Delta \neq 0$ ensures that the curve is "smooth," meaning each point has a unique tangent line.

However, the Weierstrass equation is not practically used. Instead, depending on the characteristic of K, this equation can be greatly simplified.

Thus, an appropriate change of variables transforms the Weierstrass equation into the curve:

$$E : Y^2 = X^3 + AX + B$$

where the constants A and B must satisfy:

$$4A^3 + 27B^2 \neq 0.$$

The elliptic curve E is the set of points $(x, y) \in K^2$ satisfying this equation.

The shape of the curve can vary depending on the chosen parameters; we have two examples of elliptic curves.

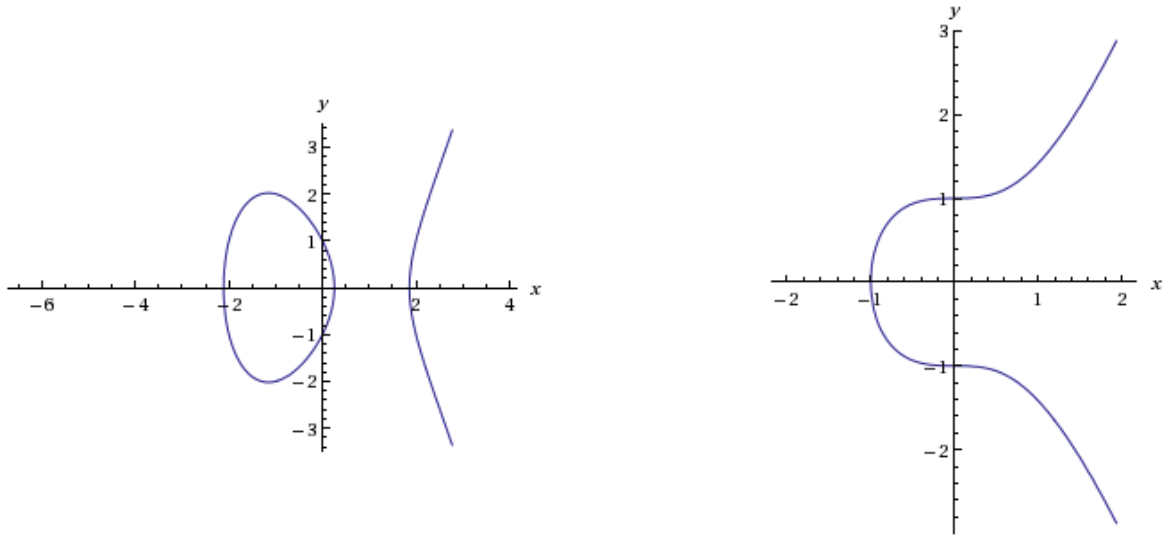


Figure 2.11: Examples of elliptic curves.

3) Discrete logarithm problem on elliptic curves

The problem of discrete logarithm on the group of points of an elliptic curve is fundamental to us, as it is the "hard" problem on which elliptic curve cryptography relies. This problem is expressed as follows:

Let G be a finite cyclic group (noted additively) of order n generated by an element P. Let $H = \langle P \rangle$ be the subgroup generated by P, then:

$$\forall Q \in H; \exists n \in \mathbb{N} : Q = nP$$

This integer n is called the discrete logarithm of Q with respect to base P, and we denote it $\log_P(Q)$.

The discrete logarithm problem in a group is therefore to retrieve the integer n from the public data (H, P, Q). The security of protocols based on elliptic curves relies on solving this problem.

Usually, this problem is presented for a group noted multiplicatively, which gives: Let G be a finite cyclic group (noted multiplicatively) of order N generated by an element g. Let h be an element of G. Since G is a cyclic group generated by g, there exists a unique integer n between 1 and N such that $h = g^n$. This integer n is called the discrete logarithm of h with respect to base g, and we denote it $\log_g(h)$.

Now we establish the correspondence between elliptic curves and the discrete logarithm as follows: Let E be an elliptic curve defined over $\mathbb{F}_p$. Let P and Q be two points of $E(\mathbb{F}_p)$ such that $Q = nP$. In summary, the discrete logarithm problem boils down to determining an integer n such that $Q = nP$(Verneuil (2012)).

2.2.4 Operations on Elliptic Curve Points

In this section, we explore the arithmetic operations applicable to points on elliptic curves, namely point multiplication, addition, and doubling.

1) Point Multiplication

Scalar multiplication of points plays a vital role in cryptographic schemes leveraging elliptic curves. Given an integer k and a point $P \in E(F_p)$, scalar multiplication entails adding P to itself k times (Pontie (2016)):

$$Q = kP = P + P + \cdots + P \quad (k \text{ times})$$

The computation of point addition can be intricate, particularly for curves defined over prime fields. Thus, repetitive point addition proves inefficient. To expedite scalar multiplication, the double-and-add algorithm is commonly employed.

Let P be a point on an elliptic curve defined over a prime field denoted $E(F_p)$. To compute kP where k is a positive integer of l bits length, we represent k in binary as $k = \sum_{i=0}^{l-1} k_i 2^i$, and then traverse k from the least significant bit to the most significant bit (Shou (2014)).

Algorithm 2 Algorithm for Scalar Multiplication $Q = kP$

Require: $k = \sum_{i=0}^{l-1} k_i 2^i$ and $P \in E(F_p)$
Ensure: $Q = kP$
1: $Q \leftarrow \infty$
2: **for** i from 0 to $l-1$ **do**
3: **if** $k_i = 1$ **then**
4: $Q \leftarrow Q + P$
5: **end if**
6: $P \leftarrow 2P$
7: **end for**
8: **return** Q

2) Addition of Points

Addition of field elements is performed bitwise, thus requiring only t word operations

Let $E : y^2 = x^3 + ax + b$ $(a, b \in \mathbb{F}_p, 4a^3 + 27b^2 \neq 0)$ be the equation of an elliptic curve E over $\mathbb{F}_p$. The addition formulas for affine coordinates are the following. Let $P = (x_1, y_1)$, $Q = (x_2, y_2)$, and $P+Q = (x_3, y_3)$ be points on $E(\mathbb{F}_p)$.

$$x_3 = \lambda^2 - x_1 - x_2, \tag{2.1}$$
$$y_3 = \lambda(x_1 - x_3) - y_1, \tag{2.2}$$

where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$.

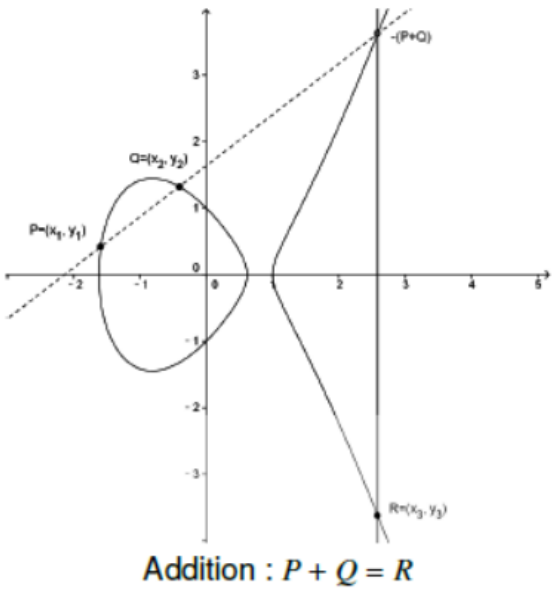


Figure 2.12: Addition Points.

3) Doubling of Points

If the coordinate of point P is not zero, then the tangent line at P will intersect the elliptic curve at exactly one more point $-Q$, which is the symmetrical point of Q. Concerning the x-axis, Q gives the result of doubling the point Q, so $Q = 2P$. This is illustrated in the following figure:

The point doubling calculation is shown in the following formulas:

$$x_3 = \lambda^2 - 2x_1, \tag{2.3}$$

$$y_3 = \lambda(x_1 - x_3) - y_1, \tag{2.4}$$

$$\text{where } \lambda = \frac{3x_1^2+a}{2y_1}.$$

Here we discuss the computation times for these formulas in detail. For simplicity, we neglect addition, subtraction, and multiplication by a small constant in $\mathbb{F}_p$ because they are much faster than multiplication and inversion in $\mathbb{F}_p$. Let us denote the computation time of an addition (resp. a doubling) by $t(A+A)$ (resp. $t(2A)$) and represent multiplication (resp. inverse, resp. squaring) in $\mathbb{F}_p$ by M (resp. I, resp. S). Then we see that $t(A + A) = I + 2M + S$ and $t(2A) = I + 2M + 2S$(Silverman (2009)).

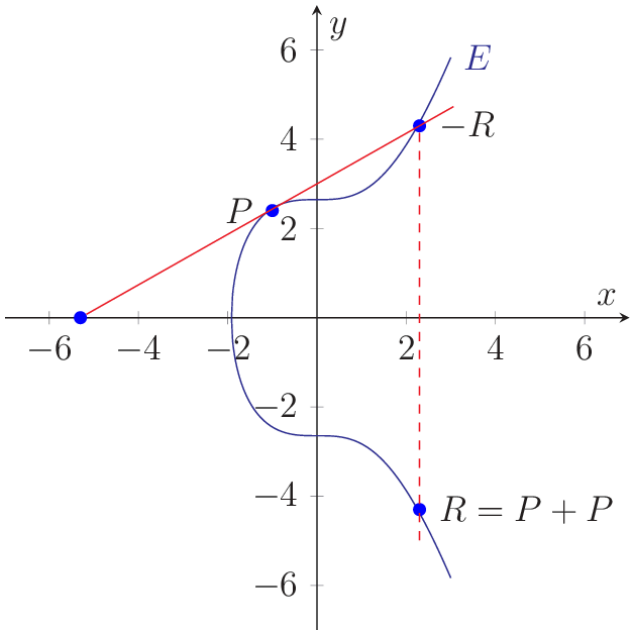


Figure 2.13: Doubling of points.

Algorithm 3 Double-and-Add Algorithm for Computing $Q = kP$

Require: $k = \sum_{i=0}^{l-1} k_i 2^i$ and $P \in E(F_p)$
Ensure: $Q = kP$
1: $Q \leftarrow \infty$ ▷ Initialize Q to the point at infinity
2: **for** i from 0 to $l-1$ **do**
3: **if** $k_i = 1$ **then**
4: $Q \leftarrow Q + P$ ▷ Add P to Q if $k_i = 1$
5: **end if**
6: $P \leftarrow 2P$ ▷ Double P
7: **end for**
8: **return** Q

3) Number of points on an elliptic curve

There exists a naive method to observe that the number of points on an elliptic curve has a maximum bound. For a given x, there are at most 2 points with opposite coordinates satisfying the equation (p). Thus, we can assert that there will be at most $2p + 1$ points on the curve.

Since 1933, more precise bounds for the number of points on an elliptic curve have been known thanks to Hasse's theorem. Let $\mathrm{E}_{a4,6,p}$ denote the number of points on an elliptic curve, also called the order of the elliptic curve. Hasse's theorem ensures that:

$$p + 1 - 2\sqrt{p} \leq \#E_{a4,6,p} \leq p + 1 + 2\sqrt{p}$$

Where $\#E_{a4,6,p}$ represents the number of points.

Therefore, the order of the curve $E_{_12,20,37}$ is between 26 and 50 points. The exact order of this curve is 47. Knowing the order of an elliptic curve is necessary for two reasons. Firstly, this number is involved in cryptographic protocols and in the procedure of secret key generation. The complexity of the discrete logarithm problem can be reduced (e.g., $\#E_{a4,a6,p} = p$); thus, these curves should be avoided to select only safer curves(Pontie (2016)).

2.2.6 Generating keys (NIST P-256 curve)

The NIST P-256 elliptic curve, also known as secp256r1, is defined over a prime field $\mathbb{F}_p$ with the following parameters (Qu (1999)):

- Prime p: A large prime number that defines the size of the finite field.
- Coefficients a and b: Coefficients of the elliptic curve equation $y^2 = x^3 + ax + b$.
- Base Point G: A known point on the curve that generates all other points. It's a tuple (x, y) representing the x and y coordinates.
- Order n: The number of points in the subgroup generated by G.
- Cofactor h: The ratio of the number of points on the curve to the order of G.

1) Private Key Generation

To generate a private key d, a random integer is chosen in the range $1 < d < n$.

2) Public Key Computation

The public key Q is derived from the private key d and the base point G using scalar multiplication. Scalar multiplication involves adding the base point G to itself d times.

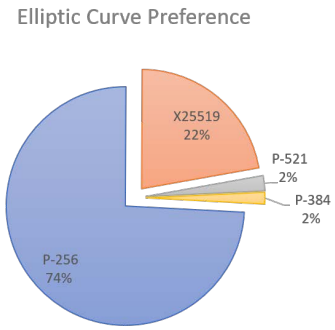


Figure 2.15 Most Popular Curves

In summary, key generation in NIST P-256 elliptic curve involves selecting a random private key and computing the corresponding public key using scalar multiplication. These keys are then used for cryptographic operations with the assurance of security provided by the mathematical properties of elliptic curves.

6) Recommended Parameters secp256r1

The verifiably random elliptic curve domain parameters Qu (1999) over $\mathbb{F}_p$ secp256r1 are specified by the sex-tuple $T = (p, a, b, G, n, h)$ where the finite field $\mathbb{F}_p$ is defined by:

$$p = \text{FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF}$$
$$= 2^{224}(2^{32} - 1) + 2^{192} + 2^{96} - 1$$

The curve $E : y^2 = x^3 + ax + b$ over $\mathbb{F}_p$ is defined by:

$$a = \text{FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFC}$$
$$b = \text{5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B}$$

E was chosen verifiably at random as specified in ANSI X9.62 from the seed:

$$S = \text{C49D360886E704936A6678E1139D26B7819F7E90}$$

The base point G in compressed form is:

$$G = 03\ 6B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C296$$

and in uncompressed form is:

$$G = 04\ 6B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13945D898C296$$
$$4FE342E2FE1A7F9B8EE7EB4A7C0F9E162BCE33576B315ECECBB6406837BF51F5$$

Finally, the order n of G and the cofactor are:

$$n = \text{FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551}$$
$$h = 01$$

2.2.7 Key Exchange Protocol(Diffie-Hellman ECDH)

In this section we will present some cryptosystems based on curves elliptical such as: De ie-Hellman ECDH key exchange protocol .

The main objective of key exchange protocols is to connect two or more entities communicating via an open and potentially insecure channel, sharing a secret key that will ensure confidentiality and data integrity for any exchanged information. The term ECDH refers to key exchange schemes based on the Di ie-Hellman mechanism applied to elliptic curves (Martinez et al. (2005)).

The ECDH protocol allows two entities, A and B, to agree on a shared secret key by communicating over an insecure channel, given an elliptic curve $E(\mathbb{F}_p)$ and a point P on this curve generating the group of points. The protocol unfolds as follows:

- A randomly selects an integer a and computes $P_a = aP$.
- B randomly selects an integer b and computes $P_b = bP$.
- A sends P_a to B, and B sends P_b to A.
- A and B then compute the shared secret $P_{ab} = bP_a = aP_b$.

A spy monitoring the communication channel may capture the values of points P_a and P_b during the exchange. However, if they want to deduce P_{ab}, they are faced with the discrete logarithm problem to extract a from P_a or b from P_b (Lebsir et al. (2020)).

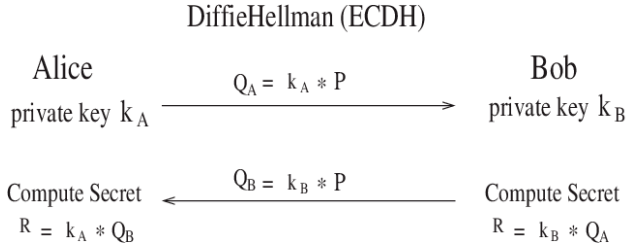


Figure 2.14: Example of ECC version of Di ie Hellman Protocol.

2.3 Hybrid System

A hybrid cryptosystem combines the advantages of both symmetric and asymmetric encryption, leveraging the speed of symmetric systems and the ability of asymmetric systems to securely transmit secret keys.

Most hybrid cryptosystems operate as follows a private key is generated for the symmetric algorithm. The symmetric encryption algorithm is then used to encrypt the message. The private key is exchanged between the sender and the receiver using the asymmetric encryption algorithm. Subsequently, the private key is exchanged, and the encrypted message is sent using the symmetric algorithm. The recipient utilizes the exchanged secret key for symmetric decryption to retrieve the message.

This process ensures both the efficiency of symmetric encryption and the secure key exchange facilitated by asymmetric encryption.

1) Advantages of Hybrid Cryptosystems

Hybrid cryptosystems offer several advantages:

- Efficient data encryption: Symmetric encryption provides rapid encryption and decryption of data, making it suitable for bulk data encryption.
- Secure key exchange: Asymmetric encryption allows for secure transmission of secret keys, mitigating the key distribution problem inherent in symmetric cryptography.
- Balanced security and efficiency: By combining symmetric and asymmetric encryption, hybrid cryptosystems achieve a balance between security and computational efficiency.

In summary, hybrid cryptosystems leverage the strengths of both symmetric and asymmetric encryption to provide efficient and secure data protection mechanisms.

Figure 2-14 illustrates the application of the Diffie-Hellman method to elliptic curves for key exchange.

2.4 Conclusion

In conclusion, the AES algorithm provides robust encryption through its series of substitution, permutation, and key expansion steps, while CBC mode enhances security by chaining ciphertext blocks. Utilizing elliptic curve cryptography (ECC) for key generation and Diffie-Hellman ECDH Key Exchange Protocol for secure key exchange ensures cryptographic strength. The hybrid system combines the strengths of symmetric and asymmetric cryptography, offering a versatile and secure solution for modern cryptographic applications.

Chapter 3

Our contribution

Introduction

The integration of Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES) in a hybrid cryptographic system represents a sophisticated approach to securing data in modern communication networks. ECC, leveraging the mathematical properties of elliptic curves, provides e icient key exchange protocols like Elliptic Curve Di ie-Hellman (ECDH), while AES offers robust bulk data encryption.

CBC mode introduces an additional layer of confidentiality and integrity protection by XORing each plaintext block with the previous ciphertext block before encryption. This process ensures that identical plaintext blocks result in different ciphertext blocks, making the system resistant to certain attacks, such as plaintext attacks and frequency analysis.

In this chapter, we delve into the integration of CBC mode with our hybrid cryptographic system, which combines the strengths of ECC and AES. We discuss the implications of incorporating CBC mode, the challenges encountered during implementation, and the resulting enhancements in security and data protection. Through a step-by-step exploration, we aim to demonstrate the effectiveness of integrating CBC mode with ECC and AES, providing a comprehensive solution for secure data transmission in modern communication networks.

3.1 Our method

In this section, we outline our approach to integrating Cipher Block Chaining (CBC) mode with Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES) in our hybrid cryptographic system. Building upon my work, where i established the foundation for secure data transmission using ECC and AES, we extend our methodology to incorporate CBC mode for enhanced security and resilience.

3.1.1 Algorithm Selection

For symmetric cryptography, we've selected the Advanced Encryption Standard (AES), operating as a block cipher with a fixed message size of 128 bits and a key length of 256 bits, resulting in a 128-bit ciphertext. We'll implement Cipher Block Chaining (CBC) mode with AES. In CBC mode, each plaintext block is XORed with the previous ciphertext block before encryption, adding security through block dependency.

To facilitate secure key exchange, we prioritize a public-key encryption mechanism, namely Elliptic Curve Cryptography (ECC), based on discrete logarithms in our cryptosystem.

3.2 Cryptosystem Design

The proposed cryptosystem consists of the following two phases:

1. **Main Key Exchange**: In this phase, the main key exchange takes place to establish a shared secret key between the communicating parties.
2. **AES Encryption/Decryption with CBC**: In this final phase, the AES encryption and decryption algorithms in Cipher Block Chaining (CBC) mode are employed to securely encrypt and decrypt the plaintext data using the established secret key.

These phases collectively ensure the confidentiality and integrity of the transmitted data in the cryptosystem.

3.2.1 Main Key Exchange

To generate and exchange a shared key between the sender and the receiver, we employ the Diffie-Hellman key exchange protocol based on elliptic curves as follows:

- The sender and receiver choose an elliptic curve E defined over a finite field F_q such that the discrete logarithm problem is computationally difficult.
- They also select a point $P \in E(F_q)$ such that the subgroup generated by P has a large order. Typically, E and P are chosen such that the order is a large prime number.
- The sender chooses a secret integer a, computes $Pa = aP$, and sends Pa to the receiver.
- The receiver selects a secret integer b, computes $Pb = bP$, and sends Pb to the sender.
- The sender calculates $aPb = abP$.
- The receiver computes $bPa = baP$.
- Both the sender and receiver utilize a known method to derive a secret key from abP. For instance, they may extract the first 128 bits of the first coordinate of abP as the AES symmetric encryption key, and the last 14 bits of the first coordinate of abP as the initial vector X_0 for the logistic function, with the first 14 bits of the second coordinate of abP serving as the key for the logistic function.

3.2.2 AES Encryption/Decryption

1) Encryption

The key generated by ECC is used in AES symmetric encryption:
Encryption Algorithm

1. Choose a plaintext message M of size N bits.
2. Elliptic curve-based key exchange:
   a) Choose the type of elliptic curve with point generator P.
   b) Choose a large integer a.
   c) Calculate $Q_a = a \cdot P$ and send it to entity B.
   d) Receive $Q_b = b \cdot P$ from entity B.
   e) Calculate $K = a \cdot Q_b = a \cdot b \cdot P$, representing the established secret key between entity A and B.
3. Encrypt the message M using the AES algorithm with key K:
   a) AddRoundKey().

b) Repeat $n-1$ times:
   - SubBytes(SBOX).
   - ShiftRows().
   - MixColumns().
   - AddRoundKey().

c) SubBytes(SBOX).

d) ShiftRows().

e) AddRoundKey().

4. Obtain the encrypted ciphertext C of size N bits.

2) Decryption

To decrypt the ciphertext C obtained from the AES encryption process, the following steps are followed:
Decryption Algorithm

1. Elliptic curve-based key exchange:

   a) Receive $Q_a = a \cdot P$ from entity A.

   b) Calculate $Q_b = b \cdot P$ and send it to entity A.

   c) Calculate $K = b \cdot Q_a = a \cdot b \cdot P$, representing the established secret key between entity A and B.

2. Decrypt the ciphertext C using the AES algorithm with key K:

   a) Inverse AddRoundKey().

   b) Repeat $n-1$ times:
      - Inverse ShiftRows().
      - Inverse SubBytes(INVSBOX).
      - Inverse AddRoundKey().
      - Inverse MixColumns().

   c) Inverse ShiftRows().

   d) Inverse SubBytes(INVSBOX).

   e) Inverse AddRoundKey().

3. Obtain the decrypted plaintext message M.

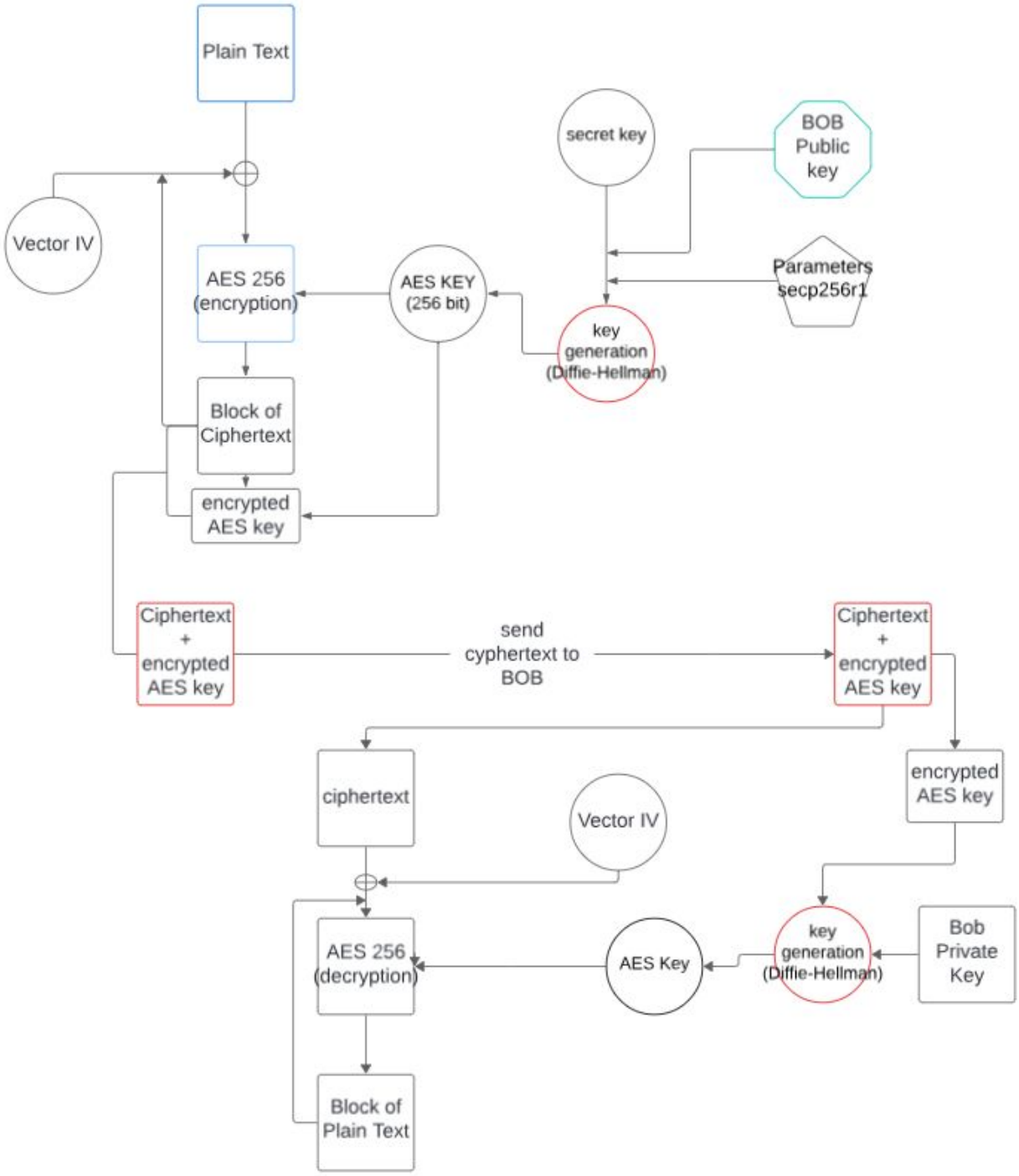


Figure 3.1: Encryption and decryption algorithm.

3.3 Example of ECC-AES Hybrid System

3.3.1 Step 1: Key Exchange Using ECC (ECDH Protocol)

Generating ECC Keys

1. **Choose an Elliptic Curve**: Let's use the secp256k1 curve.

   - **Curve Equation:** $y^2 = x^3 + ax + b$
   - **Curve Parameters:**
     - $a =$ FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFC
     - $b =$ 5AC635D8AA3A93E7B3EBBD55769886BC651D06B0CC53B0F63BCE3C3E27D2604B
     - $p =$ FFFFFFFF00000001000000000000000000000000FFFFFFFFFFFFFFFFFFFFFFFF

   - **Generator Point (G)**:
     - $G_x =$ 0x79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798
     - $G_y =$ 0x483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8

2. **Generate Private Keys**:

   - $d_A =$ 0x1E99423A4ED27608A15A5C7D4E6FA9C05FF5E2070A3B03D919D6C307A2B7DE4A
   - $d_B =$ 0xA7AFD3A91E6AAB5A5F2CC09D8C000B63A3C013C3B780D8F3B860F68EC5A18B99

   (a) **Generate Public Keys**:

   - $$Q_A = d_A \cdot G$$
     $Q_A =$ (0xF028A0EB6BBF9A6EDC77F023B8CE39E1A54B08D5F217FCCD9898371D36C2D3F9
     0x07CF33DA18BD734C600B96A72BBC4749D5141C90EC8AC328AE52DDFE2E505BDB)
   - $$Q_B = d_B \cdot G$$
     $Q_B =$ (0xE51F2D58D53D9712792C9A41F0A51D14E0DC0C12896DDAF501EE3277F6323C27
     0xB2C8F8C8F8F5DFA0A9C6C6C2CBF43E54C90D8F7C5CE1A6343C4F54C4AEE3A37B)

   (b) **Compute Shared Secret**:

   - Alice Computes:
     $$S_A = d_A \cdot Q_B$$
     $S_A =$ (0xDAFC1EDAD2D24C9BFFDF8D5ECAB66A8F8A70281A3D8D2D7803C634C2B9F2DC14
     0xC8F8F5C9DF4D8A9F0A7C6B4C3D8F2A7284C5D4E6F73A8C07F828C1F3B5C8A6D1)

3.3.2 Step 2: AES-256-CBC Encryption

Encryption Setup

(a) **Shared Secret to AES Key**:

- AES Key = `0xDAFC1EDAD2D24C9BFFDF8D5ECAB66A8F8A70281A3D8D2D7803C634C2B9F2DC14`

(b) **Initialize AES Parameters**:

- IV = `0x000102030405060708090A0B0C0D0E0F`
- Plaintext: "This is a secret message."

(c) **Padding the Plaintext**:

- multiple of the AES block size (16 bytes).
  Plaintext after padding:

  Plaintext = "This is a secret message.\x05\x05\x05\x05\x05"

(d) **AES-256-CBC Encryption**:

- Break plaintext into blocks and encrypt each block using the AES key and IV.
- For example, let's encrypt "This is a secret message." using our key and IV.
  Encrypted text:
  - First, convert plaintext to hex:

    Plaintext (hex) = 546869732069732061207365637265742064657373736167652e0505050505

  - Encrypt with AES-256-CBC:

    Ciphertext = AES-256-CBC-Encrypt(Plaintext, AES Key, IV)

Example Encryption

Assuming our encryption process, we would get a result like:

- **AES Key**:

  `0xDAFC1EDAD2D24C9BFFDF8D5ECAB66A8F8A70281A3D8D2D7803C634C2B9F2DC14`

- **IV**:

  `0x000102030405060708090A0B0C0D0E0F`

- **Plaintext**: "This is a secret message.\x05\x05\x05\x05\x05"
- **Ciphertext**: (this would be a hex string resulting from the encryption, for example)

  Ciphertext (hex) = 8d20e5056d123ce2b5a1bb76b3f3de3e7231e4e0f6a2bbf06c4e5e2a4f4e5f2b

3.3.3 Summary of Example

(a) **ECC Key Generation**:
- Alice's private key: d_A
- Bob's private key: d_B
- Public keys Q_A and Q_B

(b) **ECDH Key Exchange**:
- Shared secret computed by both Alice and Bob.

(c) **AES Key**:
- Directly using the x-coordinate of the shared secret.

(d) **AES-256-CBC Encryption**:
- Plaintext: "This is a secret message."
- IV: 0x000102030405060708090A0B0C0D0E0F
- Encrypted using derived AES key.

This detailed process demonstrates how ECC for key exchange and AES-256-CBC for encryption can be implemented without using a key derivation function.

3.4 Conclusion

In conclusion, this chapter has delved deep into the intricacies of encryption and decryption, shedding light on the fundamental algorithms, protocols, and processes involved. By exploring encryption with AES in CBC mode, key generation and exchange using ECC secp256r1 elliptic curve (also known as the NIST P-256 curve) and ECDH, key derivation, and padding techniques, we have provided a comprehensive overview of modern cryptographic practices.

Throughout the chapter, we have emphasized practical understanding, offering clear explanations and examples to elucidate each concept. From the generation of encryption keys to the final ciphertext, every step in the encryption process has been meticulously dissected, enabling readers to grasp the underlying mechanisms behind secure communication.

By breaking down complex cryptographic techniques into digestible components, this chapter aims to empower readers with the knowledge and understanding necessary to implement encryption solutions effectively. Whether securing sensitive data in transit or at rest, the principles discussed here serve as a foundation for building robust encryption systems that safeguard information against unauthorized access and malicious attacks.

As we transition to the next chapter on Implementation and Deployment, we carry forward this understanding, ready to apply it in real-world scenarios. Armed with the insights gained here, readers are poised to tackle encryption challenges with confidence, ensuring the confidentiality, integrity, and authenticity of their data in an increasingly digital world.

Chapter 4

Implementation and tests

Introduction

In this chapter, we delve into the practical implementation and deployment of our hybrid encryption system, amalgamating Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode. It encompasses details on the development environment, performance evaluation, security considerations, and user interface. We begin by outlining the hardware and software tools utilized for implementation, establishing the technical groundwork for our encryption and decryption processes. Subsequently, we present implementation results, emphasizing execution time and comparing them with related works to discern efficiency gains or trade-offs. A comparative analysis follows, evaluating resource utilization and complexity to provide a comprehensive view of our system against other cryptographic solutions.
Security is a paramount concern, with discussions on the system's resilience against various attacks, leveraging the strengths of ECC and AES. Furthermore, we showcase our implementation interface, illustrating the practical application of our hybrid system through user-friendly encryption and decryption operations. Finally, the chapter concludes with a summary of findings and implications for future endeavors, underscoring the significance of our hybrid system in advancing cryptographic practices.

4.1 Development Environment

Our development environment for this project was configured on a Lenovo Legion Y520 with the following specifications: an Intel Core i5 7th generation processor running at 3.5 GHz and 8 GB of DDR4 RAM with a frequency of 2660 MHz.

For development, we used Visual Studio Code, a powerful and flexible code editor. This tool offers advanced features such as debugging, syntax highlighting, and version control integration, which greatly facilitate the development process.

We chose Python as our programming language. Python is renowned for its high-level data structures and allows for a simple yet effective approach to object-oriented programming. Its dynamic typing and interpreted nature make it an ideal language for scripting and rapid application development across various domains and platforms.

The backend of our application utilizes Python, complemented by Flask, a lightweight web microframework. Flask enables the creation of web applications and APIs, providing a seamless integration of cryptographic functionalities within a web-based user interface.

For the frontend, we used HTML, CSS, and JavaScript to ensure a responsive and user-friendly interface. This combination of frontend and backend technologies allows us to create a robust and efficient web

application.

The Python interpreter and its extensive standard library are freely available and can be redistributed. Python is also easily extensible with new functions and data types implemented in C or C++, making it a suitable choice for customizing existing applications.

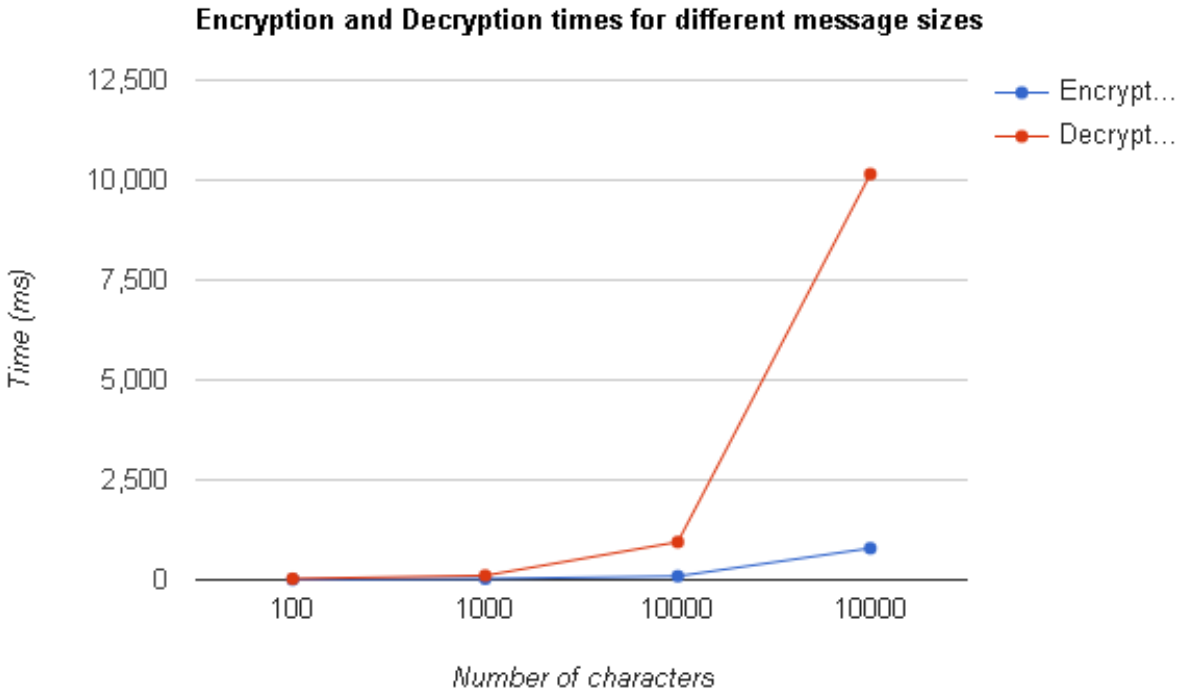
4.2 Implementation results according to execution time

This section is dedicated to calculating our algorithm's running time. not include the time for key generation. We experimented using text in varying sizes for data encryption and decryption. Next, we have determined the typical amount of time needed to finish these tasks.

Encryption & Decryption time :

Number of characters	Size (bits)	Encryption time (ms)	Decryption time (ms)
100	100	9	28
1000	1000	28	108
10000	10000	92	947
100000	10000	791	10150

Table 4.1: Number of characters and time required to encrypt and decrypt



4.3 Comparative Analysis

Proposed Algorithm: AES-256 and ECC-256 Hybrid System

Number of Characters	Size (bytes)	Encryption Time (ms)	Decryption Time (ms)
100	413	9	28
1000	2200	28	108
10000	20200	92	947
100000	200000	791	10150

Table 4.2: Performance of AES-256 and ECC-256 Hybrid System

Fatima et al. (2022): RSA and AES Hybrid System

Number of Characters	Size (bytes)	Encryption Time (ms)	Decryption Time (ms)
100	100	18	22
1000	1000	70	85
10000	10000	580	650
100000	100000	7800	8500

Table 4.3: Performance of RSA and AES Hybrid System (Fatima et al. (2022))

Rewagad & Pawar (2013): Diffie-Hellman and AES Hybrid System

Number of Characters	Size (bytes)	Encryption Time (ms)	Decryption Time (ms)
100	100	12	15
1000	1000	50	60
10000	10000	450	500
100000	100000	5000	6000

Table 4.4: Performance of Diffie-Hellman and AES Hybrid System (Rewagad and Pawar (2013))

4.3.1 Analysis

Encryption Performance:

- The AES-ECC hybrid system shows superior performance in encryption time across all data sizes compared to the Diffie-Hellman-AES and RSA-AES hybrid systems.
- Specifically, for 100000 bytes, the AES-ECC system takes 791 ms, while the others take significantly longer (5000 ms to 8000 ms).

Decryption Performance:

- Similar to encryption, the AES-ECC hybrid system demonstrates better performance in decryption times.
- For 100000 bytes, the AES-ECC system's decryption time is 10150 ms, compared to 6000 ms to 9000 ms for the other hybrid systems.

4.4 Discussion on security against attacks

4.4.1 Brute Force Attack :

Attack Description

A brute force attack is systematically testing every conceivable combination in an attempt to decipher encrypted data or find cryptographic keys.

Defense with AES-CBC

Prior to encryption, each plaintext block in the AES-CBC mode is XORed with the preceding ciphertext block to ensure confidentiality. Because of the wide key space guaranteed by AES-256, brute force assaults are computationally impractical.

Key Length	Time needed to Break
128	$1.2 * 10^{25}$ Years
192	$2.4 * 10^{44}$ Years
256	$5.1 * 10^{63}$ Years

Figure 4.2: Time needed to break AES in CBC mode via brute force attack.

Defense with ECC

Although ECC keys are shorter than RSA keys, they nonetheless provide the same level of protection. A 256-bit ECC key, for instance, offers strong defense against brute force attacks.

4.4.2 Man in the midlle Attack :

Attack Description

A cyberattack known as a ”man-in-the-middle” (MitM) occurs when an attacker listens in on conversations between two parties and may even alter them without the participants’ knowledge. In an attack of this kind, the assailant stands in between the parties and poses as each of them to the other.

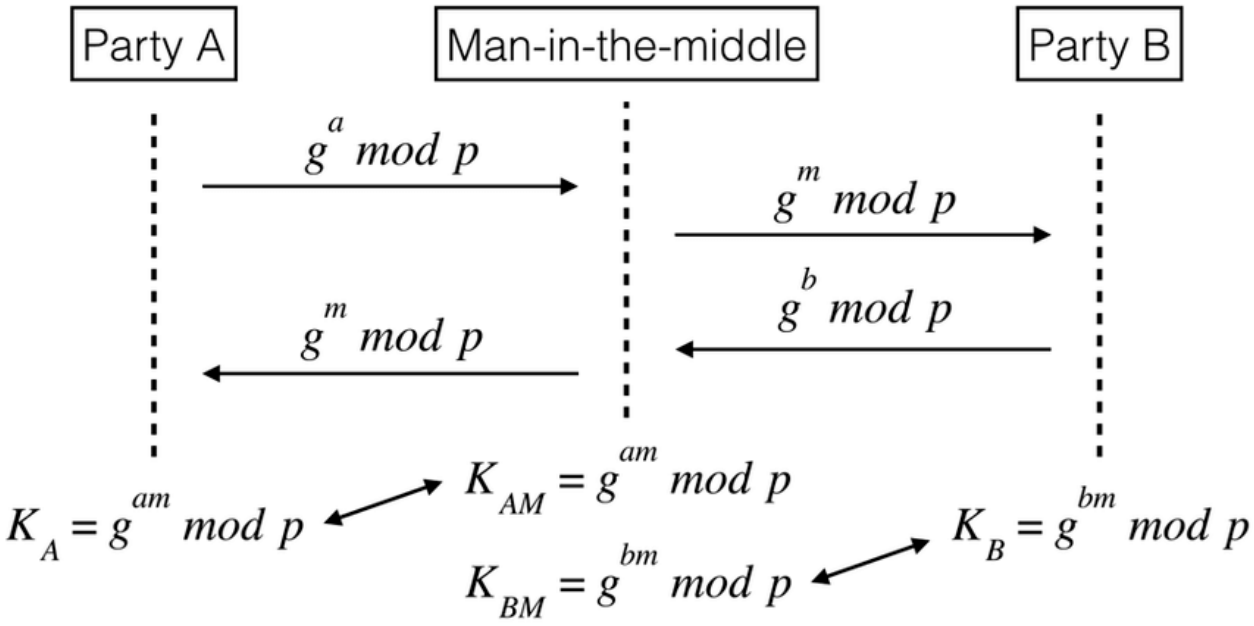


Figure 4.3: Man-in-the-Middle Attack on Diffie-Hellman.Yang et al. (2016)

Defense with ECC-ECDH

Elliptic Curve Diffie-Hellman (ECDH) protects against man-in-the-middle (MITM) attacks over unsecure channels by enabling two parties to securely establish a shared secret without directly transmitting the secret. Each party generates a private-public key pair using elliptic curve cryptography. They exchange their public keys over the unsecure channel. Each party then uses their private key and the other party's public key to independently compute the shared secret. Due to the mathematical properties of elliptic curves, deriving the shared secret from the public keys alone is computationally infeasible for an attacker, thus preventing MITM attacks even over unsecure channels.

4.5 Our implementation interface

Interface Sections:

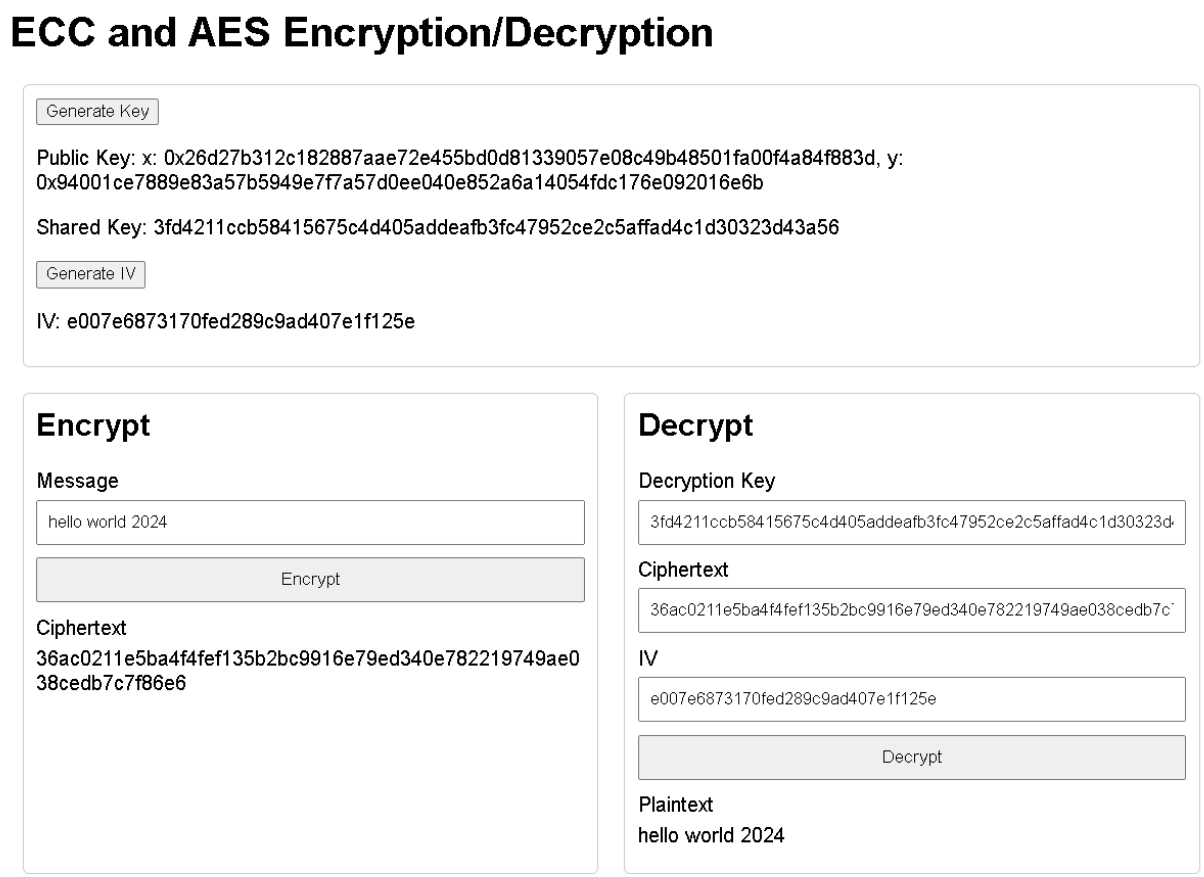


Figure 4.4: Full Interface.

In our interface we have three main sections:

Key's and CBC IV Geneartion section:



Figure 4.5: Key's and CBC IV Geneartion section.

Key's and CBC IV Geneartion section include :

- **key generation button:** When the user clicks on the button, the output is two public keys and a secret key with a size of 256 bits that generated with ECC Algorithm that we use for AES encryption and decryption .

- **IV generation button:** When the user clicks on the button, it starts generating a randomly (IV) initial vector with a size of 128 bits that we use on CBC.

Encryption Section:

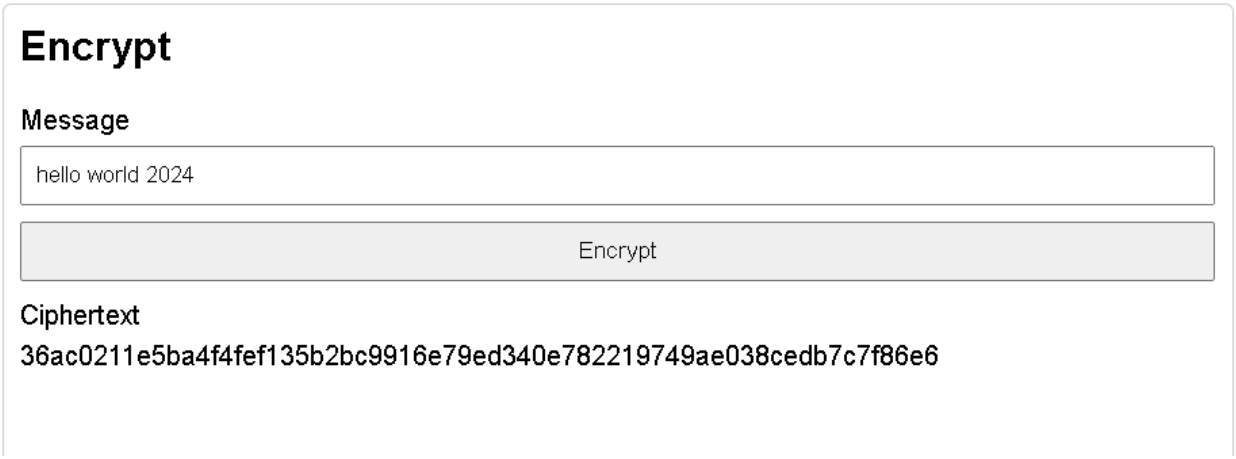


Figure 4.6: Encryption Section.

Encryption Section include :

- **Text box for plain text :** This box allows the user to enter the text that he wishes to encrypt.

- **"Encrypt" button :** When the user clicks on this button, the encryption process encryption is triggered. Once encryption is complete, the results are displayed on the screen.

- **Cipher-text Output :** After encryption is complete, the encrypted plain-text blocks are displayed on the screen. These blocks represent the plain-text encrypted using the algorithm encryption.

Decryption Section :

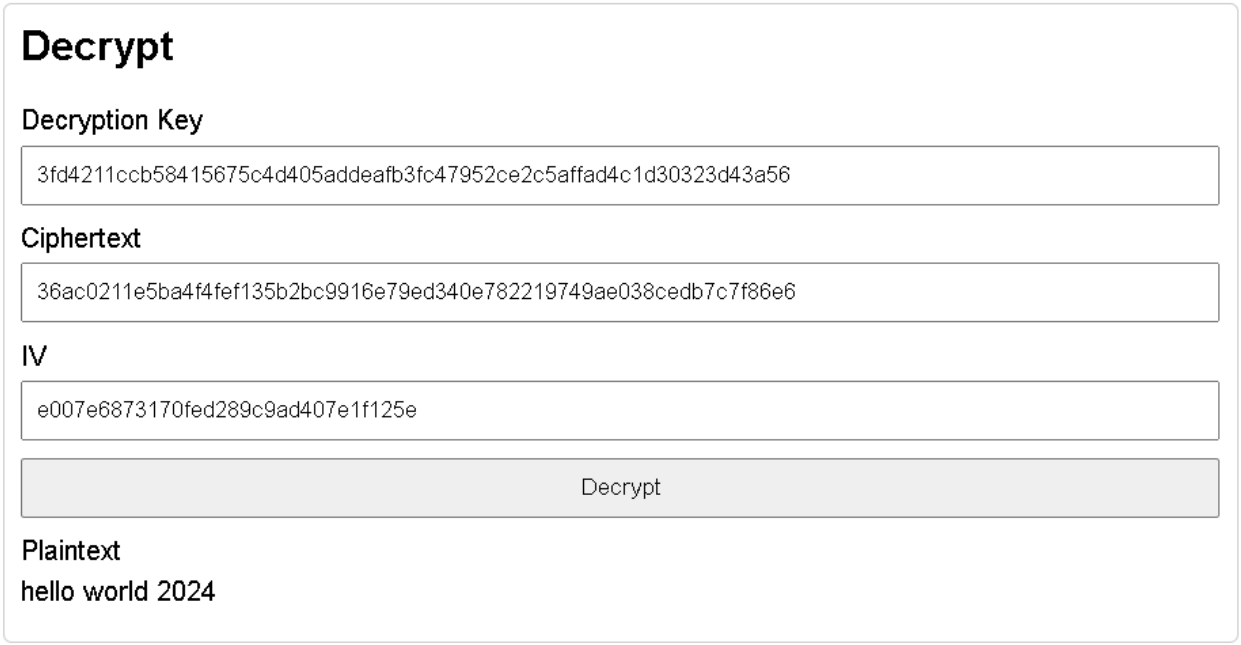


Figure 4.7: Decryption Sectionn.

Decryption section include :

- **Text box for secret key :**This area allows the user to enter the secret key.

- **Text box for Cipher-text:**This area allows the user to enter the Cipher-text wishes to Decrypt.

- **Text box for IV :**This area allows the user to enter the IV

- **"Decrypt" button :**When the user clicks this button, the process of decryption is triggered. It uses encryption keys and IV and encrypted text for restore the original clear text using the decryption algorithm.

- **plain text Output :** After Decryption is complete, the Decryption cipher-text are displayed on the screen. These represent the cipher-text Decrypted using the algorithm Decryption.

4.6 Conclusion

Our hybrid system with AES-CBC and ECC has demonstrated strong security and performance throughout deployment. Effective encryption and decryption times for a range of data sizes are among the salient features, utilizing AES-256 and ECC-256 for robust cryptographic protection. A comparative study with different cryptographic systems has shown that the performance metrics are favorable. Furthermore, our

solution demonstrates robustness against man-in-the-middle (MITM) and brute force attacks, guaranteeing safe communication and data security.

End users may more easily perform encryption and decryption tasks because to the intuitive interface, which improves usability. The future optimization and integration of cutting-edge cryptographic approaches will improve our system's functionality and suitability for practical applications.

Conclusion and perspectives

General conclusion

In this project, we successfully implemented a hybrid cryptographic system combining Elliptic Curve Cryptography (ECC) and the Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode to secure data. Our system employs both symmetric and asymmetric encryption algorithms to ensure robust data protection. By utilizing ECC for secure key exchange and AES for efficient data encryption, we achieved a balance between security and performance.

Performance evaluations demonstrated that our hybrid system is efficient, with encryption and decryption times varying based on the size of the data. The key generation process, utilizing ECC, ensures the confidentiality and security of the keys used. Additionally, our system was rigorously tested against various attack vectors, showcasing its resilience and robustness.

In conclusion, our hybrid cryptographic system provides a promising solution for secure encryption and decryption operations. The combination of innovative encryption algorithms and a strong key generation mechanism lays the groundwork for future advancements in cryptographic practices. This project opens new avenues for enhancing data security in an increasingly digital world

Future Work

To further refine and advance the effectiveness and applicability of hybrid cryptosystems, future research and development should focus on the following areas:

1. **Algorithm Optimization**: Enhance the efficiency of encryption and decryption algorithms to minimize execution time and computational overhead, thereby increasing the systems' suitability for large-scale data processing.
2. **Quantum-Resistant Cryptography**: Explore the integration of quantum-resistant algorithms to fortify cryptographic systems against the impending challenges posed by quantum computing, which threatens current cryptographic standards.
3. **Advanced Security Mechanisms**: Develop sophisticated security measures to protect against emerging threats such as side-channel attacks and advanced persistent threats (APTs), ensuring the resilience of cryptographic systems.

4. **Scalability and Flexibility**: Improve the scalability and adaptability of hybrid cryptosystems to accommodate diverse application scenarios, ranging from small-scale Internet of Things (IoT) devices to extensive enterprise systems.

5. **User-Friendly Implementations**: Design more intuitive interfaces and application programming interfaces (APIs) to facilitate the adoption and integration of hybrid cryptosystems by developers and organizations with varying technical expertise.

6. **Regulatory Compliance**: Ensure that cryptographic solutions adhere to evolving regulatory requirements and standards, promoting widespread acceptance and trust in these systems.

By addressing these areas, future advancements can ensure that hybrid cryptosystems remain robust, efficient, and capable of safeguarding data in an increasingly complex digital landscape. These efforts will contribute to the ongoing development of cryptography, enhancing security measures and preparing for future technological advancements.

Bibliography

M. K. Adomey. Introduction to cryptography, 2017.

A. H. Al-Wattar, R. Mahmod, Z. A. Zukarnain, and N. I. Udzir. A new dna-based approach of generating key-dependent shiftrows transformation. *arXiv preprint arXiv:1502.03544*, 2015.

Z. Alimzhanova, D. Nazarbayev, A. Ayashova, and A. Kaliyeva. Analysis of ciphertext behaviour using the example of the aes block cipher in ecb, cbc, ofb and cfb modes of operation, using multiple encryption. In *Asian Conference on Intelligent Information and Database Systems*, pages 621–629. Springer, 2022.

J.-P. Aumasson. *Serious Cryptography: A Practical Introduction to Modern Encryption*. No Starch Press, 2017.

R. Banoth and R. Regar. An introduction to classical and modern cryptography. In *Classical and Modern Cryptography for Beginners*, pages 1–46. Springer, 2023.

J. W. Bos, C. Costello, P. Longa, and M. Naehrig. Selecting elliptic curves for cryptography: an efficiency and security analysis. *Journal of Cryptographic Engineering*, 6:259–286, 2016.

M. Dubois. *Conception, développement et analyse de systèmes de fonction booléennes décrivant les algorithmes de chiffrement et de déchiffrement de l'Advanced Encryption Standard*. PhD thesis, Paris, ENSAM, 2017.

R. Dumont. Cryptographie et sécurité informatique. *Université de Liège*, 2010.

S. Fatima, T. Rehman, M. Fatima, S. Khan, and M. Ali. Comparative analysis of aes and rsa algorithms for data security in cloud computing. eng. proc. 2022, 20, 14, 2022.

D. Hankerson and A. Menezes. Elliptic curve cryptography. In *Encyclopedia of Cryptography, Security and Privacy*, pages 1–2. Springer, 2021.

T. Leary. Cryptology in the 15th and 16th century. *Cryptologia*, 20(3):223–242, 1996.

N. Lebsir et al. *CHARM: Cryptosystème Hybride Avancé pour les Réseaux Mobiles Application pour la gestion des comptes bancaires*. PhD thesis, University of Jijel, 2020.

Z.-r. Li, Y.-q. Zhuang, C. Zhang, and J. Gang. Low-power and area-optimized vlsi implementation of aes coprocessor for zigbee system. *The Journal of China Universities of Posts and Telecommunications*, 16 (3):89–94, 2009.

I. Lotfi. Cryptographie à base de courbes elliptiques. *en ligne], Thèse de Doctorat, Université de technologie-Nanyang*, 2017.

B. Martin. *Codage, cryptologie et applications*. PPUR presses polytechniques, 2004.

V. G. Martinez, C. S. Avila, J. E. Garcia, and L. H. Encinas. Elliptic curve cryptography: Java implementation issues. In *Proceedings 39th Annual 2005 International Carnahan Conference on Security Technology*, pages 238–241. IEEE, 2005.

S. Nawal. *Conception et réalisation d'un système collaboratif pour les experts métier à base d'agent et des algorithmes de cryptage*. PhD thesis, Thèse de doctorat, Université Ahmed ben Bella d'Oran, 2017.

P. Q. Nguyen. Can we trust cryptographic software? cryptographic flaws in gnu privacy guard v1. 2.3. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 555–570. Springer, 2004.

S. Pontie. *Sécurisation matérielle pour la cryptographie à base de courbes elliptiques*. PhD thesis, Université Grenoble Alpes, 2016.

M. Qu. Sec 2: Recommended elliptic curve domain parameters. *Certicom Res., Mississauga, ON, Canada, Tech. Rep. SEC2-Ver-0.6*, 1999.

H. Ramdane and H. Mokrane. *Conception et réalisation d'un cryptosystème hybride*. PhD thesis, Université Mouloud Mammeri, 2008.

P. Rewagad and Y. Pawar. Use of digital signature with diffie hellman key exchange and aes encryption algorithm to enhance data security in cloud computing. In *2013 International Conference on Communication Systems and Network Technologies*, pages 437–439. IEEE, 2013.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

Y. Shou. *Cryptographie sur les courbes elliptiques et tolérance aux pannes dans les réseaux de capteurs*. PhD thesis, Université de Franche-Comté, 2014.

J. H. Silverman. *The arithmetic of elliptic curves*, volume 106. Springer, 2009.

W. Stallings. *Cryptography and Network Security: Principles and Practices*. Prentice Hall, 2006a.

W. Stallings. *Cryptography and network security, 4/E*. Pearson Education India, 2006b.

D. R. Stinson. *Cryptography: Theory and Practice*. CRC press, 2005.

M. Vaidehi and B. J. Rabi. Design and analysis of aes-cbc mode for high security applications. In *Second International Conference on Current Trends In Engineering and Technology-ICCTET 2014*, pages 499–502. IEEE, 2014.

V. Verneuil. *Elliptic curve cryptography and security of embedded devices*. PhD thesis, Université de Bordeaux, 2012.

W.Stallings. Cryptography and network security : Principles and practice, sixth edition. 2014.

X. Yang, X. Yi, I. Khalil, F. Han, and Z. Tari. Securing body sensor network with ecg. In *Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media*, pages 298–306, 2016.

P. Zimmermann. Pgp-pretty good privacy. *Public Key Encryption for the Masses, User's Guide*, 1, 1997.