**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**

**Université A. Mira de Béjaïa**

**Faculté des Sciences Exactes**

**Département d'Informatique**


# Mémoire de fin de cycle

En vue de l'obtention du diplôme de **Master Professionnel en Informatique**

**Option :** Administration et Sécurité des Réseaux (ASR)

## THÈME

**Proposition d'un système hybride EKDA**
**à la base de la cryptographie à courbes elliptiques**

**Réalisé par :**

Mr AOUCHICHE Lounes

Mlle SAIDOUNE Katia

**Encadré par :**

Mme SABRI Salima

**Membres du jury :**

| | | |
|---|---|---|
| **Président :** | Mr MOKTEFI Mohand | Université A. Mira Béjaïa |
| **Examinatrice :** | Mme BATTAT Nadia | Université A. Mira Béjaïa |
| **Examinateur :** | Mr CHEKRID Mohamed | Université A. Mira Béjaïa |
| **Examinatrice :** | Mme HOUHA Amel | Université A. Mira Béjaïa |

**Promotion : 2024/2025**

# REMERCIEMENTS

Au terme de ce travail, nous souhaitons remercier toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire.

Tout d'abord, nous remercions **ALLAH**, source de force et de courage, qui nous a guidés et soutenus tout au long de ce projet.

Nos remerciements les plus sincères vont à **Madame SABRI Salima**, notre encadrante, pour avoir accepté de nous accompagner dans ce mémoire. Nous la remercions pour sa disponibilité, ses conseils précieux et son aide tout au long de la réalisation de ce mémoire. Son soutien a été essentiel pour nous permettre d'avancer.

Nous remercions également **tous les enseignants** qui ont participé à notre formation. Grâce à leurs cours et à leur engagement, nous avons acquis les connaissances nécessaires pour mener ce mémoire à bien.

Enfin, nous adressons nos remerciements aux **membres du jury** qui ont pris le temps de lire et d'évaluer notre mémoire. Leurs remarques et suggestions nous seront très utiles pour améliorer notre mémoire.

À tous, un grand merci pour votre aide, votre confiance et votre soutien tout au long de cette étape importante de notre parcours.

# DÉDICACE

*À ma très chère **mère**,*

*Quoi que je fasse ou que je dise, je ne saurai point te remercier comme il se doit. Ton affection me couvre, ta bienveillance me guide et ta présence à mes côtés a toujours été la source de ma force pour affronter les différents obstacles.*

*À mon très cher **père**,*

*Tu as toujours été à mes côtés pour me soutenir et m'encourager. Que ce travail traduise ma gratitude et mon affection.*

*À mes très chers frères **Khaled** et **Belkacem***

*pour vos soutiens moral et conseils précieux tout au long de mes études.*

*À mes **amis** et **proches***

*Et à toutes les personnes qui m'ont aimé et soutenu, qui ont cru en moi et m'ont encouragé. Que ce travail soit le reflet de votre confiance en moi et le fruit de votre confiance et de votre inspiration.*

***AOUCHICHE LOUNES***

# DÉDICACE

*À mes chers **parents**,*
*Pour leur amour, leurs sacrifices et leur soutien inestimable, sans qui rien de tout cela n'aurait été possible.*

*À mon cher frère **Amer**,*
*Pour sa présence rassurante et ses encouragements constants.*

*À mes **sœurs** bien-aimées,*
*Pour leur tendresse, leur écoute et leur réconfort dans les moments de doute.*

*À mes **amis** sincères,*
*Pour leur précieuse amitié et leur soutien tout au long de ce parcours.*

*Enfin, à toutes **celles** et **ceux** qui, de près ou de loin,*
*ont contribué à faire de ce mémoire bien plus qu'un simple travail académique.*
*une véritable étape de vie marquée par votre présence.*
*Je vous en suis profondément reconnaissante.*

***SAIDOUNE KATIA***

# TABLE DES MATIÈRES

# TABLE DES FIGURES

# LISTE DES TABLEAUX

# Liste des algorithmes

# LISTE DES ABRÉVIATIONS

**A**

| | |
|---|---|
| 3DES | Triple Data Encryption Standard |
| AES | Advanced Encryption Standard |
| AES-CCM | Advanced Encryption Standard - Counter with CBC-MAC |
| AES-GCM | Advanced Encryption Standard - Galois/Counter Mode |

**C**

| | |
|---|---|
| CPU | Central Processing Unit |

**D**

| | |
|---|---|
| DES | Data Encryption Standard |
| DoS | Denial of Service |

**E**

| | |
|---|---|
| ECC | Elliptic Curve Cryptography |
| ECDH | Elliptic Curve Diffie-Hellman |
| ECDSA | Elliptic Curve Digital Signature Algorithm |

**F**

| | |
|---|---|
| FIPS | Federal Information Processing Standards |

**H**

| | |
|---|---|
| HKDF | HMAC-based Key Derivation Function |

**I**

| | |
|---|---|
| IKM | Input Keying Material (dans le contexte HKDF) |

**M**

| | |
|---|---|
| M-LWE | Module Learning With Errors |

## N

| | |
|---|---|
| N/A | Not Applicable |
| NIST | National Institute of Standards and Technology |

## O

| | |
|---|---|
| OKM | Output Keying Material (dans le contexte HKDF) |

## P

| | |
|---|---|
| PRK | Pseudo-Random Key (dans le contexte HKDF) |

## R

| | |
|---|---|
| RAM | Random Access Memory |
| RSA | Rivest-Shamir-Adleman |

## S

| | |
|---|---|
| SHA | Secure Hash Algorithm |

# INTRODUCTION GÉNÉRALE

À l'ère numérique, la sécurité de l'information repose de plus en plus sur des mécanismes cryptographiques sophistiqués. Notamment, la cryptographie sur courbes elliptiques (ECC) qui est utilisée comme une solution de choix, alliant efficacité et haut niveau de sécurité. Néanmoins, l'avènement des ordinateurs quantiques menace les fondements mêmes de ces algorithmes classiques, rendus vulnérables à de nouvelles formes d'attaques, notamment celles basées sur l'algorithme de Shor.

Face à ces défis, une question se pose avec acuité : comment concevoir un système cryptographique capable de résiste aux menaces actuelles et émergentes, tout en préservant des performances acceptables en termes de rapidité et de consommation de ressources ?.

L'objectif principal de ce mémoire est de proposer une solution cryptographique hybride combinant les avantages de plusieurs approches. Nous présentons un système basé sur l'ECC (cryptographie sur courbes elliptiques), Kyber (cryptosystème résistant aux attaques quantiques), Dilithium (schéma de signature post-quantique) et AES-GCM (mode de chiffrement symétrique authentifié). Afin de garantir à la fois la sécurité des échanges et l'authenticité des données dans un contexte où les menaces coexistent. Pour atteindre cet objectif, nous avons adopté une méthodologie expérimentale consistant à :

- Étudier les fondements théoriques de la cryptographie classique et post-quantique ;
- Implémenter un système hybride intégrant les algorithmes susmentionnés ;
- Évaluer les performances (temps d'exécution, consommation RAM, utilisation CPU) du système proposé.

Ce mémoire est structuré comme suit :

- **Le premier chapitre** présente les généralités sur la cryptographie, en mettant en lumière ses objectifs fondamentaux, ses principales familles d'algorithmes, ainsi que les différentes typologies d'attaques cryptographiques.

- **Le deuxième chapitre** est consacré à la cryptographie sur courbes elliptiques (ECC), en abordant ses fondements théoriques, ses différentes classifications, ainsi que ses principales applications cryptographiques.
- **Le troisième chapitre** constitue la partie applicative du mémoire. Il détaille la conception, la mise en œuvre et l'évaluation d'un système cryptographique hybride basé sur ECC, Kyber, Dilithium et AES-GCM.

Ce mémoire se clôt par une conclusion générale récapitulant les principales réalisations, tout en ouvrant la voie à des améliorations futures et à de nouvelles perspectives de recherche.

Chapitre 1

# Fondements et État de l'art de la Cryptographie

## 1.1 Introduction

La cryptographie est l'art et la science de la sécurité informatique. Elle joue un rôle crucial dans la protection des données sensibles face aux menaces croissantes liées à la transmission et au stockage d'informations. Face à cette problématique, il devient indispensable de comprendre les fondements de cette discipline.

Dans ce chapitre, nous présentons les notions de base de la cryptographie, son origine, ses principaux concepts, ainsi que les objectifs de sécurité, les différentes attaques. Enfin, nous abordons les techniques utilisées pour renforcer la confidentialité des échanges.

## 1.2 Étymologie et définition de la cryptographie

Le terme « cryptographie » vient de deux mots grecs « **kryptós** » signifiant caché ou secret, et « **gráphein** », qui signifie écrire. Littéralement, la cryptographie est l'art de cacher l'information pour qu'elle soit incompréhensible sauf au destinataire, elle désigne l'ensemble des techniques qui permettent de chiffrer les messages [1].

## 1.3 Terminologie

Les principaux termes utilisés dans la cryptographie sont [2] :

**Cryptographie :** Est l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un rapport donné.

**Crypto-système :** Un crypto-système est constitué d'un algorithme cryptographique ainsi toutes les clés possibles et tous les protocoles qui le font fonctionner.

**Cryptanalyse :** Opposée à la cryptographie, elle a pour but de retrouver le texte en clair à partir de textes chiffres en déterminant les failles des algorithmes utilises.

**Cryptologie :** C'est une science mathématique regroupant la cryptographie et la cryptanalyse.

**Texte clair :** Est le message a chiffré.

**Chiffrement :** La fonction permettant de transformer une donnée (texte, message,etc) afin de la rendre incompréhensible par une personne autre que celui qui a créé le message et ainsi que le destinataire.

**Déchiffrement :** La fonction permettant de retrouver le texte clair à partir du texte chiffré.

**Crypter :** Synonyme de chiffrer.

**Clé :** Est un paramètre utilisé en entrée d'une opération cryptographique.

**Algorithme :** Est une suite d'opérations logiques ou mathématiques, utilisés pour chiffrer des messages, vérifier l'identité d'un utilisateur.

**Protocole :** Ensemble de règles et de procédures qui définissent comment plusieurs systèmes échangent les informations de manière sécurisé.

## 1.4 Classifications de la cryptographie

La cryptographie est classifiée en des grandes familles selon leurs évolutions historiques. le schéma 1.1 illustre la hiérarchie de la cryptographie :

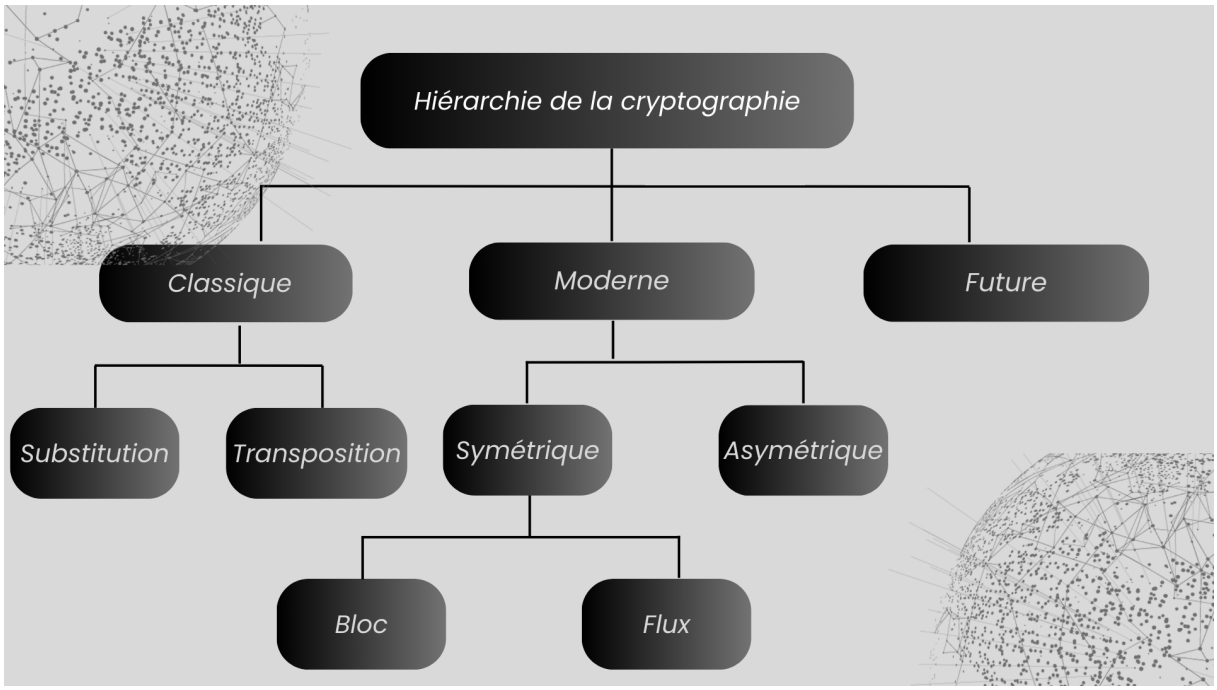



FIGURE 1.1 – Classification de la cryptographie.

### 1.4.1 Cryptographie classique

Elle est conçue avant la création des ordinateurs, désigne les méthodes anciennes de chiffrement, elle repose sur des techniques simples souvent appliquées manuellement ou à l'aide des machines mécaniques [3]. Elle est classifiée en deux types selon la façon dont les données sont traitées :

#### 1.4.1.1 Chiffrement par substitution

Elle sert à modifier les lettres ou les groupes de lettres d'un message selon une méthode [4]. Parmi ces méthodes on trouve :

- **Chiffrement de césar :** Est l'un des plus anciens systèmes cryptographie par substitution connu par Jules César. Il est utilisé dans l'armée romaine. Son principe fonctionne par décalage des lettres de l'alphabet de n positions [5], par exemple décalage de 3 positions où chaque lettre est décalée d'un certain nombre de positions dans l'alphabet.

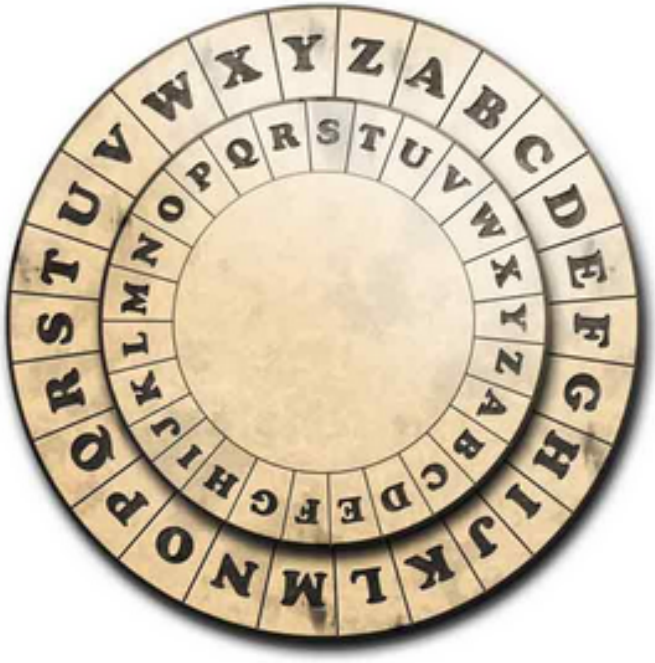


FIGURE 1.2 – Cryptage de César [5].

- **Chiffrement de vigénère :** Est une technique de chiffrement par substitution, qui utilise une clé qui définit le décalage pour chaque lettre du message (26 dans sa version original), la grande force de cette technique est que la même lettre peut être chiffré de différents manières [6].

  **Exemple :**

  **Clair :** UN TEXTE CHIFFRE AVEC VIGENERE.

  **Clé :** SE CRETS ECRESTE CRET SECRTSE.

  **Chiffré :** MR VVBMW GJZJYI CMIV NMIVRXJI

- **Chiffrement de hill :** Est un chiffrement par substitution. Dans sa version conventionnelle, l'algorithme opère sur des groupes de lettres appartenant à un alphabet de 26 lettres, par un système d'équations linéaires [7].
- **Machine engima :** C'est une machine électromécanique inventé par Arthur Scherbius repose sur la cryptographie par substitution, utilisée par les Allemands pendant la seconde guerre mondiale pour chiffrer des communications militaire [8].

**Description :**

-Chaque lettre est remplacée par une autre, l'astuce est que la substitution change d'une lettre à l'autre.

-La machine est alimentée par une pile électrique.

-Quand on appuie sur une touche du clavier, un circuit électrique est fermé, et une lampe s'allume qui indique quelle lettre codée l'on substitue.

FIGURE 1.3 – Machine Engima [9].

#### 1.4.1.2 Chiffrement par transposition

Est une technique de la cryptographie classique qui sert à réarranger les lettres (caractères) d'un message initial et de rendre les données visuellement inintelligibles, Ainsi, toutes les lettres du message initial sont conservés, mais disposées dans un ordre différent [10].

- **Transposition simple par colonne :** Sert à écrire le message horizontalement dans une matrice prédéfinie, et pour retrouver le texte chiffre, on lit la grille verticalement. le procédé inverse représente le procédé de déchiffrement [11].
- **Transposition complexe par colonne :** Un mot clé secret est utilisé pour dériver une séquence de chiffres commençant par 1 et finissant par le nombre de lettres composant le mot clé. Cette séquence est obtenue en numérotant les lettres du mot

clé en partant de la gauche vers la droite et en donnant l'ordre d'apparition dans l'alphabet.Une fois la séquence de transposition obtenue, on chiffre en écrivant d'abord le message par lignes dans un rectangle, puis on lit le texte par colonnes en suivant l'ordre déterminé par la séquence [12].

- **Transposition carre-polybique :** Un mot clé secret est utilisé pour construire un alphabet dans un tableau, permettant d'extraire les coordonnées des lignes et des colonnes correspondant aux lettres du texte à chiffrer. Ainsi, chaque lettre du texte en clair est représentée par deux chiffres écrits verticalement sur deux lignes. L'étape qui suit, consiste à concaténer les deux lignes obtenues précédemment pour obtenir une seule ligne de chiffres, puis a recombiner ces chiffres deux par deux. Ces nouvelles combinaisons de chiffres représentant les coordonnées de lignes et de colonnes de texte chiffré [12].

### 1.4.2 Cryptographie moderne

Elle repose sur les fondements mathématiques solides et des algorithmes plus complexes, après l'avènement des ordinateurs [3]. Deux grandes familles dominent cette approche selon le partage de clé entre les parties communicantes :

#### 1.4.2.1 Cryptographie symétrique (à clé secret)

Est une méthode de chiffrement qui utilise une seule clé partagée pour chiffrer et déchiffrer les données. Elle est basée sur une succession de transposition et de substitutions complexes des valeurs du message. Cela signifie que l'expéditeur et le destinateur doivent possèdent la même clé secret et la garder confidentielle [13].Elle est classée selon la manière dont elles traitent les données :

- **Chiffrement par bloc :** Est l'un des plus grandes catégories de chiffrement moderne symétrique. Il consiste à découper les données en blocs de taille fixe. Ces blocs ensuite chiffrés les uns après les autres selon les modes d'opérations qui définissent comment les données sont traités. Parmi les principaux modes opératoires de chiffrement par blocs : Electronic Code Book (ECB), Cipher Block Chaining (CBC), Output Feedback (OFB) et Cipher Feedback (CFB) [14].
- **Chiffrement par flux :** Est également appelé chiffrement par flot (Stream Cipher), est une méthode cryptographique moderne symétrique où les données sont traités de longueurs arbitraires bit par bit ou octet par octet sans avoir à les découper [14].

Parmi les algorithmes les plus utilisées et les plus connus dans la cryptographie symé-

trique on trouve :

- **DES (Data Encryption Standard) :** Est un algorithme de chiffrement à clé secret développé par IBM en 1977 et fut adopté par le NIST. Il est basé sur la fonction de feistel, et opère selon un chiffrement par blocs de 64 bits de texte en clair en utilisant la clé de 56 bits [8].
- **3DES (Triple DES) :** La vulnérabilité des DES à une attaque exhaustive a encouragé les chercheurs a développé d'autres algorithmes plus sûrs. Étant donnée les investissements dans les implémentations de logiciels et de processeurs dédiés à DES, on se contente d'appliquer les DES trois fois consécutives avec deux clés différentes. L'utilisation de trois étapes permet de doubler la longueur effective de la clé à 112 bits [15].
- **AES (Advanced Encryption Standard) :** Est un algorithme de chiffrement symétrique publié par le NIST en 2000. Il utilise des blocs de données de 128 bits et des clés de longueur variables (128, 192, 256 bits).Les principaux objectifs de cet algorithme étaient de remplacer l'algorithme DES après avoir mis en évidence certains aspectes vulnérables de cet algorithmes. Cet algorithme est composé de plusieurs étapes à savoir [16] :

  **Permutation :** un bloc de données de 16, 24 ou 32 octets sont permutés ensuite placés dans une matrice.

  **SubBytes :** Opération consiste à substituer chaque élément de la matrice vie une SBox.

  **Shiftrows :** Cette étape implique un décalage à gauche sur les éléments de la matrice.

  **MixColumns :** En effectuant une opération mathématique sur chaque colonne de la matrice de données et mettant le résultat dans une nouvelle matrice.

  **AddroundKey :** Cette étape consiste à faire un XOR entre la matrice qui contient la clé et le bloc de données.
- **RC4 (Rivest Cipher 4) :** Est un chiffrement par flux crée en 1987 par Ron Rivest. Il s'agit de chiffrer les données bit par bit ou octet par octet, à l'aide d'un flux de clés pseudo-aléatoireé [17].

  **Description**

  Cet algorithme fonctionne sur les octets. Ainsi, la clé, de longueur variable, peut avoir une taille comprise entre 1 et 256 octets (de 8 à 2048 bits). Elle est utilisée pour initialiser un vecteur S de 256 octets. A tout moment, S contient une permutation de toutes les cellules le composant [6].

La figure 1.4 illustre le principe du RC4 :

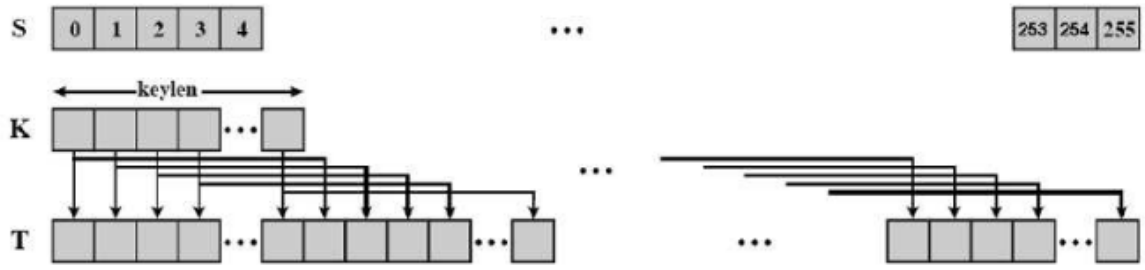



FIGURE 1.4 – Fonctionnement de chiffrement RC4 [17].

#### 1.4.2.2 Cryptographie asymétrique (à clé publique)

Est une méthode de chiffrement dans laquelle chaque utilisateur dispose d'une paire de clés, où chaque paire est composée d'une clé publique pour chiffrer le message et d'une clé privée pour le déchiffrer. Elle se base sur des fonctions à sens unique. Cela veut dire que les données cryptés avec la clé publique ne peuvent être décryptés que s'il on possède la clé secrète. Ça signifie que même si l'on a obtenu la clé publique on ne pourra pas déchiffrer les informations [18].
Nous allons maintenant présenter quelque algorithmes asymétriques les plus utilisés :

- **RSA (Rivest-Shamir-Adleman) :** Est le premier système à clé publique a été proposé par Rivest, Shamir et Adleman en 1978, fondé sur l'utilisation de l'exponentiation modulaire et la difficulté de factoriser des grands nombres [19].

L'algorithme RSA est illustré comme suite :

**Algorithm 1** Algorithme RSA

**Entrée :** $p$ et $q$ deux grands nombres premiers distincts.

**Sortie :** Clé publique $(e, N)$, clé privée $(d, N)$.

**Phase de génération des clés**

Calculer $N \leftarrow p \times q$.

Calculer l'indicatrice $\phi(N) \leftarrow (p-1)(q-1)$.

Choisir un exposant $e$ tel que $1 < e < \phi(N)$ et $\gcd(e, \phi(N)) = 1$.

Calculer $d \equiv e^{-1} \pmod{\phi(N)}$. ▷ d est l'inverse modulaire de e.

**Phase de chiffrement**

Pour un message de taille quelconque, découper un message $M$ en blocs $m_i < N$.

Appliquer à chaque $m_i$ : $c_i \leftarrow m_i^e \bmod N$.

**Phase de déchiffrement**

Sert à retrouver le message original à partir des blocs de texte crypté $c_i$.

Appliquer à chaque $c_i$ : $m_i \leftarrow c_i^d \bmod N$.

**Phase de signature numérique**

Calculer l'empreinte $h \leftarrow H(M)$ . ▷ $H$ : fonction de hachage.

Signer le haché h : $S \leftarrow h^d \bmod N$.

**Phase de vérification de signature**

Recalculer le haché à partir du message reçu $h' \leftarrow H(M)$.

Déchiffrer le message signé avec la clé publique $V \leftarrow S^e \bmod N$. **if** $V == h'$ **then**

**Retourner** "Signature valide". **else**

**Retourner** "Signature invalide".

- **ELGamal :** Est une méthode de cryptographie à clé publique inventé par Taher ELGamal en 1985. Sa sécurité repose sur la difficulté de calculer le logarithme discret [8].Cet algorithme est composé de trois étapes :

  **Génération des clés**

  -Choisir un grand nombre premier **p** et deux nombres **a** et **g** tel que : a,g <p

  -Calculer

  $$A = g^{(a)} \ \ mod \ \ p \tag{1.1}$$

  -La clé publique est **(A,g, p)** et la clé privée est **"a"**

**Chiffrement**

Pour chiffrer un message **M**, il faut procéder comme suit :

-Choisir un nombre aléatoire **b**, tel que **b<a** et le **PGCD (b,p-1) =1**.

$$B = g^{(b)} \ \ mod \ \ p \tag{1.2}$$

-Ensuite, calculer :

$$C = M \times A^{(b)} \ \ mod \ \ p \tag{1.3}$$

-Le message chiffré est alors (B,C).

**Déchiffrement**

Pour décrypter le message, on calcule :

$$M = C \times B^{(p-a-1)} \ \ mod \ \ p \tag{1.4}$$

- **ECC (Elliptic Curve Cryptography) :** Est une méthode Cryptographique moderne à clé publique proposé dans les années 1980 par Neal Kobitz et Victor Miller. Elle repose sur les propriétés mathématiques dans lequel les calculs sont effectués, comme le nom indiqué, sur une courbe elliptique [6].
  La cryptographie à courbe elliptique utilise des équations sous forme :

$$y^2 = x^3 + ax + b \tag{1.5}$$

  Cette forme repose sur la difficulté du problème du logarithme discret sur les courbes elliptiques **(ECDLP)** (voir 2.4).

### 1.4.3 Cryptographie post-quantique (future)

C'est un ensemble d'algorithmes et de méthodes cryptographiques conçues pour résister aux attaques d'ordinateurs quantiques, qui menacent de casser les systèmes classiques basés sur la factorisation des grands nombres ou du logarithme discret. Elle se base sur des techniques mathématiques (polynôme multivarié, codes, fonction de hachage, etc) qui restent difficile à inverser, même pour les ordinateurs quantiques, ce que permet de garantir la sécurité à long terme [20].

## 1.5 Objectifs fondamentaux de la cryptographie

La cryptographie vise à satisfaire plusieurs objectifs essentiels pour la sécurité de l'information. Ces objectifs, également appelés principes de sécurité, constituent les piliers sur lesquels reposent les systèmes de protection des données.
La figure 1.5 illustre les objectifs de la cryptographie :


FIGURE 1.5 – Objectif du cryptographie.

### 1.5.1 Confidentialité

Il s'agit de la garantie que des tiers non autorisés n'ont pas d'accès à des informations confidentielles et sensibles. Cela implique que les informations ne sont accessibles qu'aux personnes autorisées [15].

### 1.5.2 Intégrité

Un mécanisme qui permet de protéger un message contre toute tentative de modification par un tiers. Le destinataire doit vérifier que le message n'a pas été altéré durant son transfert. L'intégrité des données est assurée par l'application de fonction de hachage selon la procédure HMAC [15].

### 1.5.3 Authentification

C'est un mécanisme consiste à s'assurer que seules les personnes autorisées aient accès aux ressources, données, appareils, etc [21].

### 1.5.4 Non-répudiation

Elle garantie qu'une transaction ne peut être niée. Ce qui est important pour la traçabilité et la responsabilité [21].

### 1.5.5 Disponibilité

Il s'agit de la continuité de service à tout moment pour les utilisateurs autorisés. Cela implique de prévenir les interruptions, pannes ou attaques qui pourrait rendre les ressources indisponibles [15].

### 1.5.6 Contrôle d'accès

C'est un moyen technique qui détermine qui est autorisé à accéder à quelles ressources et dans quelles conditions, afin de donner la possibilité de modifier ou d'agir sur une ressource quelconque [15].

### 1.5.7 Authenticité

C'est une mécanisme consiste à vérifier que les informations proviennent bien de la source déclarée et qu'elles sont fiables [15].

## 1.6 Typologie des attaques cryptographiques

Une attaque est une tentative malveillante, menée par des individus ou des groupes, visant à accéder, perturber, Endommager ou voler des informations sensibles via des systèmes informatiques ou des réseaux.De nombreux attaquants cherchent à détruire ou briser les mécanismes de protection mis en place. Il est donc essentiel de connaître les différents types d'attaques pour concevoir des systèmes plus robustes [22].

### 1.6.1 Classification selon la position de l'attaquant

Elle désigne toutes actions en fonction de l'endroit où se situe l'attaquant par rapport au système ciblé, notamment en tant qu'observateur externe ou un acteur interne compromis [22].

#### 1.6.1.1 Attaque interne

Elle est initiée par une entité dans le périmètre de sécurité. Une entité autorisée à accéder aux ressources du système mais qui les utilise d'une manière non approuvée par ceux qui ont accordé l'autorisation [23].

#### 1.6.1.2 Attaque externe

Elle est initiée depuis l'extérieur du périmètre, par un utilisateur non autorisé ou illégitime du système [23].

### 1.6.2 Classification selon le comportement de l'attaquant

Elle désigne la manière dont les attaquants agissent, qu'elle soit passif ou actif.

#### 1.6.2.1 Attaque active

Il s'agit des interférences directs avec le système cryptographique. Les attaquants peuvent modifier, falsifie ou injecter des données dans des systèmes pour manipuler ou obtenir un accès non autorisé [23]. Parmi les attaques les plus courantes :

- **Rejeu (Replay) :** L'entité malveillante capture des transmissions de données valides et les retransmet ou les modifies légèrement pour tromper le destinataire et les faire accepter comme légitimes [24] .
- **Hameçonnage (Phishing) :** Technique basée sur l'ingénierie sociale visant à tromper une victime pour qu'elle divulgue des informations sensibles comme des identifiants de connexion, des numéros de carte bancaire ou d'autres données personnelles [25].
- **Déni de service (Denial of service, DOS) :** Vise à rendre un service indisponible, en empêchant les utilisateurs légitimes d'y accéder [25].
- **Rançongiciel (Ransomware) :** Un logiciel malveillant qui prend en otage les données d'une victime en les chiffrant , rendant ainsi les fichiers inaccessibles. Les attaquants exigent ensuite une rançon, souvent en crypto monnaie comme le bitcoin, en échange de la clé de déchiffrement permettant de récupérer les données [26].
- **Injection (Sql) :** Cyberattaque où l'acteur malveillant insère de code qui permet de récupérer, manipuler ou détruire des informations sensibles contenues dans des bases de données SQL . Ces attaques consistent à insérer des commandes spécialisées dans les champs de requête SQL. Une fois exécutées, ces commandes peuvent permettre aux acteurs malveillants d'usurper l'identité d'utilisateurs légitimes [27].
- **Homme au milieu (Man in the middle) :** Technique dans laquelle une tierce partie accède aux communications entre deux autres parties , sans qu'aucune de

ces deux parties ne s'en rende compte afin d'intercepter, d'espionner, de modifier ou de détourner les informations échangées, sans que les victimes ne s'en aperçoivent [28].

#### 1.6.2.2 Attaque passive

Ce sont des attaques où l'attaquant se met en écoute non autorisée, en surveillant simplement la transmission ou la collecte d'informations. L'oreille indiscrète n'apporte aucun changement aux données ou au système [29]. Parmi les attaques passive on trouve :

- **Force brute (bruteforce) :** Consiste à tester, l'une après l'autre, chaque combinaison possible d'un mot de passe ou d'une clé pour un identifiant donné afin de se connecter au service ciblé . Il s'agit d'une méthode ancienne et répandue chez les pirates. Le temps nécessaire à celle-ci dépend du nombre de possibilités, de la vitesse que met l'attaquant pour tester chaque combinaison et des défenses qui lui sont opposées [30].
- **Dictionnaire (dictionary attack) :** Une technique de cyberattaque qui vise à deviner un mot de passe en testant des combinaisons tirées d'une liste prédéfinie . Ces listes, appelées « dictionnaires de mots de passe », contiennent des mots, phrases ou combinaisons courantes comme « azerty », « 123456 », . . . etc [31].
- **Reniflement (Sniffing) :** Est une méthode permettant d'intercepter et d'analyser les données qui circulent sur un réseau en utilisant des logiciels appelés "sniffers", ou analyseurs de paquets, capables de capturer tout ou partie du trafic réseau et surtout Lorsqu'un réseau n'est pas sécurisé ou que les données ne sont pas chiffrées [31].
- **Surveillance par des caméras :** Elle représente une menace facilitée par des failles techniques et un usage illégal comme un enregistrement vidéo invisible qui peut être dissimulé dans un objet de la vie courante afin d'espionner le ciblé [32] .

## 1.7 Techniques avancées de sécurisation

Pour mieux comprendre la sécurisation de données, On distingue deux sections :

### 1.7.1 Mécanismes cryptographique fondamentaux

Afin de répondre aux objectifs de sécurité de la cryptographie, plusieurs mécanismes techniques et mathématiques ont été développés. Parmi ces mécanismes on peut

citer :

- **Mots de passe :** Est une suite de caractères (lettres, chiffres, symboles) utilisée pour vérifier l'identité d'un utilisateur lors de l'accès à un service (compte, ordinateur,..etc) qui doit être tenue secrète pour éviter qu'une entité non autorisée puisse accéder à une ressource ou un service [33].
- **Certificat électronique :** Document numérique qui contient toutes les cordonnées d'un interlocuteur utiles pour communiquer avec d'autres, ainsi sa clé publique. Les certificats électroniques sont utilisés principalement pour assurer l'authentification [8].
- **Signature numérique :** Mécanisme cryptographique qui permet d'authentifier l'expéditeur d'un message ou document haché en utilisant la clé privé de signataire.le destinataire vérifie cette signature avec la clé publique du signataire [34].
- **Facteur biométrique :** Est une méthode d'identification basée sur certains données humains, physiques ou comportementales, qui peuvent aller de l'empreinte digitale en passent par la voix ou la forme de la main. Elle permet d'assurer l'authentification d'une entité [6].
- **Tiers de confiance :** Un tiers de confiance peut être une personne, organisation ou système technique. Il désigne une entité neutre ou reconnue chargée de garantir la fiabilité entre deux parties ne se font pas mutuellement confiance, en se basent sur des mécanismes techniques et juridiques [8].
- **Fonction de hachage :** Est une application calculable qui transforme un message clair de longueur quelconque en un message de longueur fixe inférieur à celle de départ, appelé «empreinte de hachage » ou « haché ». Ce dernier est utilisé pour résumer une donnée de manière unique [8].

### 1.7.2 Protocoles cryptographiques

c'est un ensemble des règles et procédures utiliser dans les mécanismes de sécurité pour un objectif d'assurer la sécurité des communications ou du stockage des données. Il existe plusieurs protocoles importants comme :

- **IPsec (Internet Protocol Security) :** C'est un ensemble de règles conçu pour sécuriser les communications au niveau de la couche réseau. Il est composé de deux éléments principaux [8] :
  - **AH (Authentication header) :** Fournit l'intégrité et l'authentification des données.

  - **ESP (Encapsulating security payload) :** Fournit le chiffrement, intégrité et l'authentification des données.

- **PGP (Pretty Good Privacy) :** C'est une méthode de cryptage conçu pour garantir la confidentialité, l'intégrité et l'authenticité des communications et des informations virtuelles basé sur une méthode hybride qui combine des techniques de cryptographie à clé symétrique et à clé publique.Il est utilisée principalement pour protéger les courriels (e-mails) et les fichiers [35].
- **SSH (Secure Shell) :** Est à la fois un programme d'informatique et un protocole de communication sécurisé. Il impose une connexion sécurisé entre le client et le serveur sur un réseau non sécurisé comme internet. Le protocole SSH a été conçu pour l'objectif de remplacer les différents programmes à savoir rlogin, telnet et rsh [36].
- **TLS (Transport Layer Security) :** Protocole cryptographique pour une communication dans un réseaux informatique, notamment sur Internet, standardisé par l'IETF. TLS est un successeur de **SSL (Secure Socket Layer)** permet d'établir un canal sécurisé entre un client et un serveur, en chiffrent les données échangées afin de garantir la confidentialité, leur intégrité et l'authentification des parties impliqués [36].
- **Kerboros :** Est un système d'authentification sécurisé à tierce personne de confiance, conçu pour les réseaux TCP/IP. Il est basé sur l'utilisation de la cryptographie à clé privé.
  L'authentification est négociée par le biais d'un tiers de confiance **"Key Distribution Center (KDC)"** [8].

## 1.8 Conclusion

Dans ce chapitre, nous avons abordés de poser les bases nécessaire à la compréhension des concepts fondamentaux de la cryptographie. Dans ce qui suit comme deuxième chapitre, nous allons désormais nous intéresser à une approche moderne : la cryptographie sur les courbes elliptiques et ses différentes techniques.

Chapitre 2

# Cryptographie sur les Courbes Elliptiques (ECC)

## 2.1 Introduction

Les courbes elliptiques jouent aujourd'hui un rôle central en cryptographie et sont largement utilisées dans divers protocoles et applications de sécurité.Ce chapitre est consacré à la présentation de la cryptographie sur les courbes elliptiques (ECC).

Au premier lieu, nous commencerons par introduire les fondements mathématiques de cette cryptographie, en définent les opérations fondamentales des courbes elliptiques, les opérations algébriques associées, ainsi que le problème du logarithme discret elliptique. Une classification des principales courbes elliptiques.

Dans le second temps, nous nous intéresserons aux principales fonctionnalités cryptographiques offertes par l'ECC, notamment à travers les protocoles ECDH, pour l'échange sécurisé de clés, et ECDSA, pour la signature numérique. Enfin, un aperçu des travaux récents intégrant l'ECC dans des approches hybrides sera présenté, afin de mettre en lumière les tendances actuelles et les perspectives de recherche dans ce domaine.

## 2.2 Fondements des Courbes Elliptiques (EC)

Les fondements des courbes elliptiques reposent sur des notions mathématiques qui permettent des opérations sûres en cryptographie.

**Opérations fondamentales**

Les points d'une courbe elliptique définis sur un corps fini forment un groupe abélien sous une opération d'addition de points [6]. Les opérations fondamentales sont :

- **Addition de points**

  Soient deux points de courbe $P = (x_1, y_1)$ et $Q = (x_2, y_2)$, l'addition de deux points est calculer comme suite :

$$\lambda = \frac{y_2 - y_1}{x_2 - x_1} \mod p \tag{2.1}$$

$$x_3 = \lambda^2 - x_1 - x_2 \mod p \tag{2.2}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \mod p \tag{2.3}$$

Le résultat de doublement de point est :

$$P + Q = (x_3, y_3) \tag{2.4}$$

- **Doublement de point**

  Quand **P= Q**, on appelle double de point c'est-à-dire **P+ Q = 2P**.

  Formule :

$$\lambda = \frac{3x_1^2 + a}{2y_1} \mod p \tag{2.5}$$

$$x_3 = \lambda^2 - 2x_1 \mod p \tag{2.6}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \mod p \tag{2.7}$$

  Le résultat est donnée sous forme d'un couple :

$$2P = (x_3, y_3) \tag{2.8}$$

- **Point neutre**

  Représente l'élément neutre de l'addition sur une courbe elliptique. Il joue le même rôle que zéro dans l'addition classique P + O = P [6].

  **Description**

  Soit E(x,y) une courbe elliptique définie par l'équation

$$y^2 = x^3 - x + 1 \tag{2.9}$$

  - La première partie de figure 2.1 illustre l'addition de point comme suite :
    - Tracer la droite qui passe par les points P et R sur la courbe elliptique.
    - La droite coupe la courbe elliptique en un troisième point, que l'on appelle -Q.
    - Le point P + R = Q est le symétrique du point -Q par rapport au axe des abscisses.

  - La deuxième partie de figure 2.1 illustre le doublement de point comme suite :
    - Trouver la tangente d'un point P sur une courbe elliptique (la tangente est la droite qui touche la courbe en un seul point).
    - La tangente coupe la courbe en un point supplémentaire -Q.
    - Le point 2P est le symétrique du point supplémentaire -Q par rapport au axe

des abscisses.

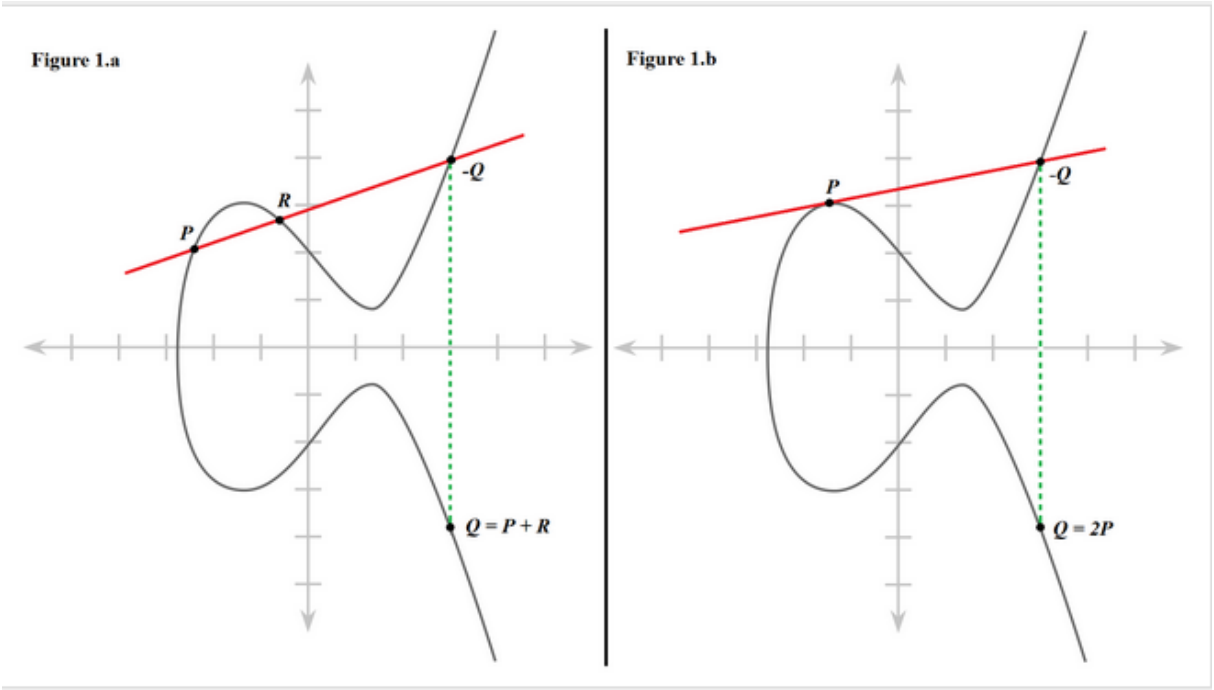



FIGURE 2.1 – Addition et doublement de point

- **Multiplication scalaire**

  Est une technique mathématique consiste à multiplier un point **P** appartenant à une courbe **E**, par un entier positif **k**, afin d'obtenir un point **Q** sur la même courbe. Elle est définie comme une répétition de l'addition de points [37].

$$Q = kP \tag{2.10}$$

  Cela signifie que l'on ajoute le point P lui-même k fois :

$$kP = \underbrace{P + P + \cdots + P}_{k \text{ fois}} \tag{2.11}$$

  La multiplication scalaire est une opération au cœur du fonctionnement cryptographique basée sur les courbes elliptiques, car elle :

  - Garantir la sécurité grâce à l'ECDLP (voir la sous-section 2.4).
  - Génère des clés publiques à partir des clés privées : $Q = kP$.
  - Génère la signature (ECDSA),(voir la sous-section 2.6.2).

- **Algorithme Double-and-add :** C'est une méthode conçue pour faciliter le calcul de la multiplication scalaire dans la cryptographie sur les courbes elliptiques [38].

**Algorithm 2** Algorithme Double-and-Add pour la multiplication scalaire.

**Entrée :** un entier $k$ (scalaire), un point $P$ sur une courbe elliptique.
**Sortie :** Le point $Q = kP$.
**Phase d'initialisation**
Convertir $k$ en binaire : $k = (b_n b_{n-1} \dots b_0)_2$.
Initialiser $Q \leftarrow \mathcal{O}$ .
**Phase de calcul (Double-and-Add)**
**for** $i$ *de* 0 *à* $n$ **do**
    Parcours des bits de $k$.
**if** $b_i = 1$ **then**
    $Q \leftarrow Q + P$ (Addition ).
**else**
    $Q \leftarrow 2Q$ (Doublement) .
**Retourner** $Q$

## 2.3 Classification des courbes elliptiques

Les courbes elliptiques peuvent se présenter sous différentes formes et structures, selon le contexte mathématique ou cryptographique dans lequel elles sont utilisées.

### 2.3.1 Corps de définitions

Elle représente la nature du corps mathématique sur lequel une courbe elliptique est définie. On distingue [6] :

- **Corps réels :** les courbes elliptiques sont définies sur les nombres réels. Elles prennent une forme continue dans le plan, et sont souvent représentées graphiquement sous forme de courbes symétriques par rapport à l'axe des abscisses.
- **Corps complexes :** elles correspondent plutôt à une surface en forme de tore (comme un anneau). Elle ne représente plus une simple courbe elliptique sur un plan.
- **Corps finis :** un corps fini, noté Kp (où p est un nombre premier), ou (corps binaire), est un ensemble fini d'éléments dans lequel les opérations d'addition, de soustraction, de multiplication et de division (sauf par zéro) sont bien définies et

vérifient les propriétés algébriques usuelles. Ils constituent le fondement principal de leur utilisation en cryptographie moderne.

### 2.3.2 Usage cryptographique

Les courbes elliptiques sont définies et classées selon leurs performances, leurs niveaux de sécurité et leurs compatibilités avec des implémentations matérielles ou logicielles.
Le tableau 2.1 illustre les tailles des clés exigées par des courbes standardisées, ainsi que le nombre d'opérations nécessaire pour casser le système [39].

| Courbe | Taille de clé | Niveau de sécurité |
|---|---|---|
| **Secp384r1** | 384 | $2^{192}$ |
| **Secp521r1** | 521 | $2^{256}$ |
| **Secp256r1** | 256 | $2^{128}$ |
| **Curve25519** | 25 | $2^{128}$ |
| **Secp224r** | 22 | $2^{112}$ |
| **Secp192r1** | 19 | $2^{96}$ |

TABLE 2.1 – Tableau comparatif de différentes courbes elliptiques.

## 2.4 Elliptic Curve Discrete Logarithm Problem (ECDLP)

Problème de logarithme discret elliptique constitue la base de la sécurité des systèmes cryptographiques utilisant les courbes elliptiques. Il consiste à retrouver un entier "**a**" à partir de deux points P et Q dans une courbe E. Bien que l'opération **"a.P"** (appelée multiplication scalaire) soit facile à effectuer, l'opération inverse (trouver a) est extrêmement difficile à réaliser [40].
Toute la robustesse des algorithmes ECC repose sur l'impossibilité pratique de résoudre le problème ECDLP, ce qui montre l'efficacité des courbes elliptiques et leur impacts sur la cryptographie.

## 2.5 Génération des clés ECC

Dans la cryptographie à courbes elliptiques, la génération des clés repose sur les opérations mathématiques sur des points appartenant à une courbe choisie, ce qui permet de produire les deux clés publique et privée [6].

- **Clé privée :** C'est un entier **d** choisi généralement,

$$1 \leq d \leq n - 1 \tag{2.12}$$

avec n est l'ordre de point générateur G.

- **Clé publique :** Elle dépend de la clé privée. Elle est calculée par la multiplication scalaire, comme suit :

$$Q = d \cdot G \tag{2.13}$$

avec G est un point de base sur la courbe.

## 2.6 Fonctionnalités cryptographiques de l'ECC

La cryptographie sur courbes elliptiques offre plusieurs fonctionnalités essentielles à la sécurisation des communications numériques, notamment l'échange de clés et la signature numérique.

### 2.6.1 Échange de Clés ECDH

Le protocole Elliptic Curve Diffie-Hellman (ECDH), initialement proposé par MILLER [41] et KOBLITZ [42], permet l'établissement sécurisé d'une clé secrète partagée via un canal non sécurisé.

### Fonctionnement du protocole ECDH

La Figure 2.2 illustre le processus de fonctionnement du protocole ECDH :

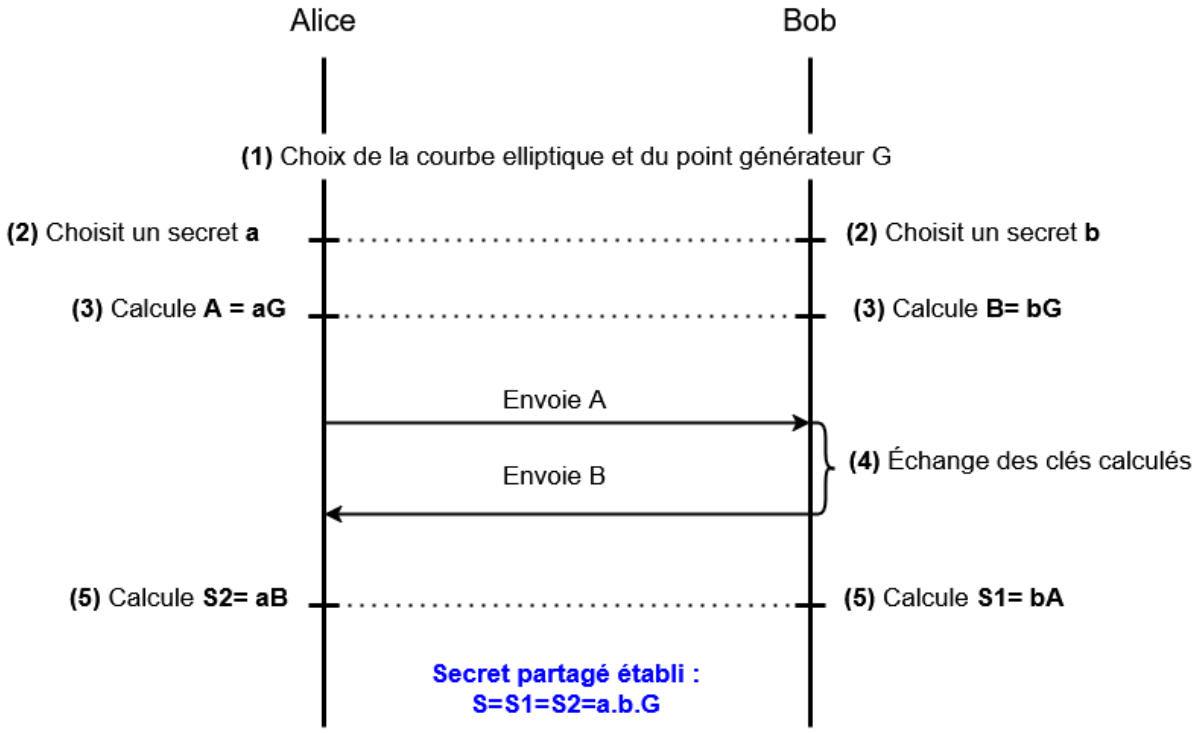



FIGURE 2.2 – Processus de génération de clés ECDH.

### Description

1. **Initialisation des paramètres :** la première étape est la sélection d'une courbe elliptique $E$ et d'un point générateur $G$ d'ordre premier $n$ .
2. **Génération des clés privées :** après avoir sélectionner une courbe, l'étape qui suit consiste à choisir des aléa pour les deux entités Alice et Bob comme étant des clés privées, tel que
$$a, b \leftarrow \{1, \ldots, n-1\} \tag{2.14}$$
3. **Calcul des clés publiques :** les clés publiques sont obtenues par les opérations suivantes :
$$A = a.G \quad \text{(Alice)}. \tag{2.15}$$
$$B = b.G \quad \text{(Bob)}. \tag{2.16}$$
4. **Échange des clés publiques :** l'étape que suit le calcul des clés publique est la Transmission de $A$ et $B$ via le canal public c-à-d
-Alice envoie A vers Bob.
-Bob envoie B vers Alice.

5. **Calcul du secret partagé :** Chaque entité doit calculer son secret correspondent :
   -Alice calcule :

$$S = a.B = a(b.G) \tag{2.17}$$

   -Bob calcule :

$$S = b.A = b(a.G) \tag{2.18}$$

   **Résultat :** consiste à trouver un secret commun entre ces entités.

   avec : **S=a.(b.G)=b(a.G)**

### 2.6.2 Signatures Numériques ECDSA

L'algorithme Elliptic Curve Digital Signature Algorithm (ECDSA), permet de produire des signatures numériques assurant l'authentification des messages [43].

#### 2.6.2.1 Processus de Signature ECDSA

Afin d'assurer l'intégrité, l'émetteur doit signer son message pour qu'il l'envoie au partie récepteur.

La figure 2.3 illustre le processus de signature d'un message par ECDSA :

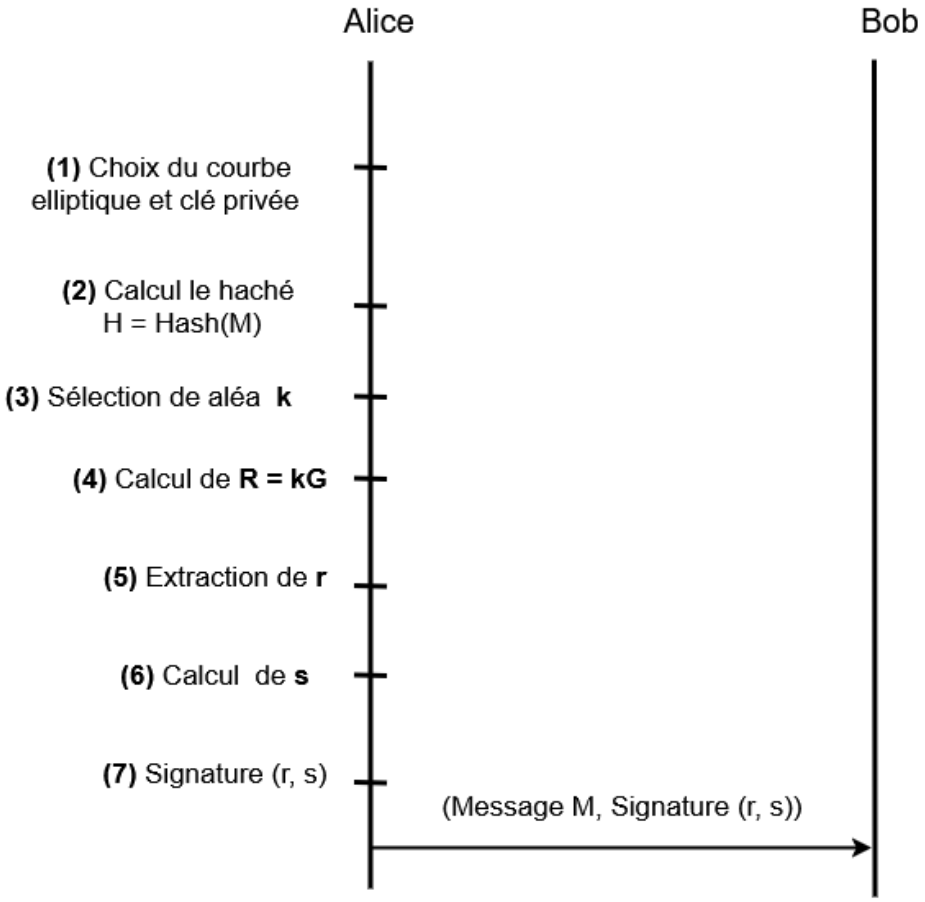



FIGURE 2.3 – Processus complet de génération de signature ECDSA.

**Description**

1. **Choix de la courbe elliptique**

   La première itération consiste à choisir une courbe elliptique.

2. **Calcul du Haché "H"**

   En utilisant la fonction SHA-256, afin de produire un condensé **"H"**.

$$H = SHA - 256(M). \tag{2.19}$$

   avec M est le message en clair.

3. **Choix d'un aléa (nonce) "K"**

   Le nonce est un entier choisi aléatoirement de tel sorte qu'il soit secret et unique pour chaque signature, Il doit être dans l'intervalle suivante :

$$k \leftarrow [1, n-1] \tag{2.20}$$

   avec n est l'ordre du point générateur G dans E.

4. **Calcul du point intermédiaire "R"**

   Le point R est calculé selon l'opération suivante :

$$R = k.G = (x_R, y_R) \tag{2.21}$$

5. **Extraction de "r"**

   La composante $x_R$ du R est utilisée pour calculer la première partie de la signature : $r = x_R$.

6. **Calcul de la valeur "s"**

   La seconde composante de la signature est la variable "s", elle est obtenue via l'équation suivante :

$$s = k^{-1} \cdot (H + d \cdot r) \mod n \tag{2.22}$$

   Avec :

   - $d$ est la clé privée de l'émetteur.
   - $k^{-1}$ est l'inverse modulaire calculé via l'algorithme d'Euclide étendu.
   - Si s=0, le processus est également répétée avec un nouveau k.

7. **Signature finale (r,s)**

   La signature d'un message représenter sous forme d'un couple $(r, s)$. Ce couple permet, à partir de la clé publique correspondante de vérifier l'authenticité et l'intégrité du message signé.

#### 2.6.2.2 Vérification de la Signature ECDSA

La figure 2.4 présente le processus de vérification de signature :

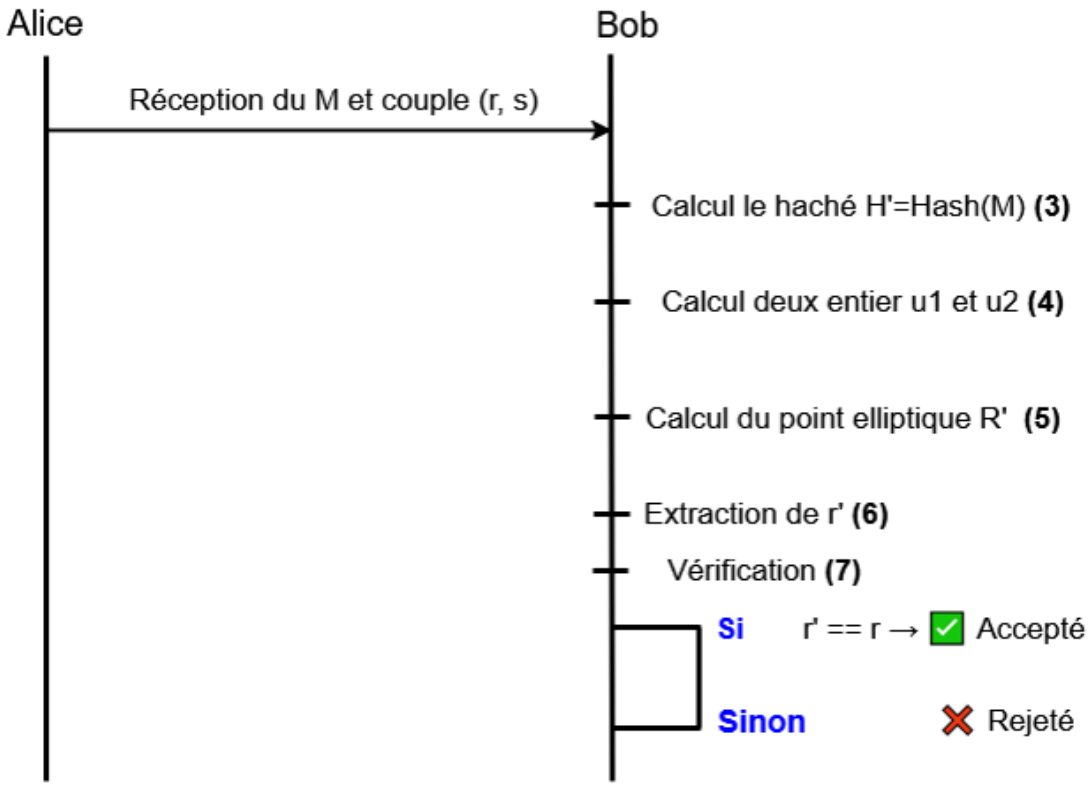



FIGURE 2.4 – Processus de vérification ECDSA.

**Description**

Á la réception, le destinataire doit recevoir les informations suivants :

- un message **M** : message initial (original).
- la signature (r,s) : couple de la signature calculer par l'émetteur.

Le processus de vérification se déroule comme suite :

1. **Vérifie les composants**

   Il vérifie d'abord que les deux entiers appartiennent à l'intervalle valide.

$$1 \leq r, s \leq n - 1 \tag{2.23}$$

2. **Calcul le Haché "H'"**

   H' = SHA-256(M)  (Même fonction de hachage que le signataire).

Cela garantit que le résumé utilisé lors de la vérification est identique à celui utiliser lors de la signature.

3. **Calcul de l'inverse modulaire "w"**

   Á partir de l'algorithme d'Euclude etendu on obtient l'inverse modulaire de s, tel que :

$$w = s^{-1} \mod n \tag{2.24}$$

   L'entier "w" est ensuite utilisé pour pondérer les termes dans l'étape suivante.

4. **Calcul des coefficients "$u_1$"et "$u_2$"**

   Á partir de H',r et s on calcule les deux entiers tel que :

$$u_1 = H' \cdot s^{-1} \mod n \tag{2.25}$$

$$u_2 = r \cdot s^{-1} \mod n \tag{2.26}$$

5. **Calcul du point elliptique "R'"**

   Afin d'obtenir un point R'. On effectue de combinaison linéaire sur la courbe elliptique, à l'aide des scalaire $u_1$,$u_2$ et du générateur G et de la clé publique Q :

$$R' = u_1 \cdot G + u_2 \cdot Q = (x'_R, y'_R) \tag{2.27}$$

6. **Extraction de "r'"**

   Á partir de R', on extrait la coordonnée $x'_R$, et on calcule :

$$r' = x'_R \tag{2.28}$$

7. **vérification**

   On compare le "r'" obtenu avec la valeur originale (reçue) "r" de la signature :

   Si **r'=r**, alors la signature est valide ✓

   Sinon, elle est invalide ×

## 2.7 État de l'art des solutions de sécurisation de la messagerie par ECC

Cette section présente un état de l'art des approches de sécurisation des systèmes de messagerie reposant sur la cryptographie à courbes elliptiques (ECC),en mettant en lumière les principaux travaux connexes menés dans ce domaine.

Dans l'article [44], Abdullah Ahmad Alhaj met en avant les avantages d'ECC pour sécuriser les données. Il montre que l'ECC, avec ses clés plus courtes, est plus efficace que RSA, surtout dans les environnements à ressources limitées comme l'IoT ou les appareils mobiles. L'auteur explique les protocoles ECDSA et ECDH, qui assurent l'intégrité et l'authenticité des données. Il compare l'ECC à d'autres algorithmes (RSA, AES, ChaCha20, SHA-256) selon plusieurs critères : taille des clés, rapidité, résistance aux attaques, confidentialité persistante (forward secrecy) – une propriété qui garantit que même si une clé privée est compromise, les communications passées restent protégées –ainsi que la capacité d'accord de clés sécurisé, qui permet à deux parties de créer une clé secrète commune sans qu'elle soit interceptée, même si la communication est surveillée. L'étude conclut que l'ECC est plus performante et mieux adaptée aux besoins actuels, notamment dans les domaines sensibles comme la finance et la santé.

Dans l'article [45], Gayathri et al proposent une solution hybride combinant AES et ECC pour renforcer la sécurité des systèmes de chiffrement. Leur approche utilise AES-192, un chiffrement symétrique avec une clé de 192 bits et 12 tours (c'est-à-dire 12 transformations successives des données effectuées à l'aide d'une clé de 192 bits), pour chiffrer les données rapidement, et ECC-256, une cryptographie asymétrique avec une clé de 256 bits, pour sécuriser l'échange de la clé AES et éviter les attaques lors de la transmission. Cette combinaison améliore la résistance aux attaques de 30 à 40 % par rapport à AES-128, qui utilise une clé de 128 bits et 10 tours. Les auteurs envisagent d'évoluer vers AES-256/ECC-384, avec des clés plus longues (256 bits pour AES, 384 bits pour ECC), afin de mieux répondre aux menaces futures.

Dans l'article [46], Abhijit Mitra et al. proposent un protocole de chiffrement des e-mails basé sur l'ECC, afin de surmonter les limites des systèmes asymétriques classiques comme RSA. Leur méthode utilise les courbes elliptiques définies sur des corps finis binaires ($GF(2^m)$) pour convertir les messages en points, puis les chiffrer à

l'aide d'opérations mathématiques spécifiques à l'ECC. Ce système offre une sécurité équivalente à RSA, mais avec des clés beaucoup plus courtes (160 bits au lieu de 1024), ce qui améliore la rapidité et réduit les ressources nécessaires. Il résiste aussi mieux aux attaques par canaux auxiliaires "attaques qui ne s'attaquent pas directement à l'algorithme, mais qui exploitent des informations physiques dégagées par un appareil lorsqu'il chiffre ou déchiffre des données" . Toutefois, le protocole pose encore des défis, notamment en termes de calculs complexes et de gestion des clés. Les auteurs suggèrent d'y intégrer des signatures numériques et d'optimiser les paramètres de courbe pour améliorer les performances.

Dans l'article [47], Neelima, CH. et Suneetha, CH. proposent une amélioration du chiffrement ElGamal basé sur les courbes elliptiques (ECC). Leur objectif est de réduire la taille du texte chiffré, qui est généralement doublée dans les versions classiques, tout en ajoutant des signatures numériques pour garantir l'authentification et l'intégrité des données. Leur méthode utilise les polynômes de Chebyshev (des fonctions mathématiques utiles pour renforcer la sécurité) afin de générer des valeurs aléatoires, utilisées pour rendre chaque chiffrement unique, imprévisible et plus difficile à attaquer. Elle permet également de réduire la taille du texte chiffré pour qu'elle soit proche de celle du message original, et d'appliquer des rotations binaires pour mieux dissimuler les données chiffrées.Les résultats montrent que cette méthode résiste bien aux attaques courantes contre les courbes elliptiques, notamment celles qui tentent de résoudre le problème du logarithme discret(ECDLP),ainsi qu'aux analyses visant à casser le chiffrement,tout en protégeant contre les attaques actives et passives.Cette solution améliore la confidentialité, l'intégrité et l'efficacité du chiffrement, et ouvre des perspectives intéressantes pour des applications nécessitant une forte sécurité avec une contrainte de taille, telles que la messagerie sécurisée.

Dans l'article [48], B.S. Murthy et al. proposent une solution pour améliorer la sécurité et les performances du chiffrement des e-mails, face aux limites des méthodes traditionnelles comme RSA, qui nécessitent des clés de grandes taille et consomme des ressources (CPU,RAM). Pour résoudre ce problème, ils utilisent la cryptographie à courbes elliptiques (ECC).Leur méthode fonctionne facilement avec des systèmes déjà utilisés pour sécuriser les e-mails, comme S/MIME (protocole sert à chiffrer et signer les e-mails) et PGP (un système qui protège les e-mails et les fichiers). ECC améliore ces systèmes en rendant le chiffrement plus rapide et plus efficace, tout en gardant un très bon niveau de sécurité.Ils s'appuient aussi sur des outils informatiques comme TensorFlow (un logiciel de machine learning créé par Google) et NumPy (une bibliothèque Python pour faire des calculs et manipuler des données).Les tests montrent que l'ECC chiffre les données très rapidement, en seulement 0,05 milliseconde, alors que RSA prend beaucoup plus de temps. En plus, ECC offre une sécurité solide. Cette solution pourrait être utilisée dans plus de systèmes de communication et aider à mieux se protéger contre les futures menaces, comme celles venant de l'informatique quantique.

Dans l'article [49], Yang et al. proposent une solution pour sécuriser les systèmes de messagerie instantanée, qui permettent aux utilisateurs d'échanger en temps réel des messages, fichiers ou contenus multimédias. Ces systèmes, souvent vulnérables aux interceptions, manquent de protections solides. Pour y remédier, les auteurs présentent le protocole SIMPP (Secure Instant Messaging and Presence Protocol), qui utilise ECC, une méthode moderne offrant une sécurité élevée avec des clés courtes. SIMPP combine la signature ECDSA , ECDH, AES-128 et HMAC-SHA-256 (une fonction de hachage qui vérifie que les messages n'ont pas été modifiés, en combinant une clé secrète et le hachage SHA-256). Ce protocole atteint le même niveau de sécurité que RSA avec des clés six fois plus petites, ce qui le rend plus rapide et mieux adapté aux appareils comme les smartphones. De plus, il reste compatible avec XMPP/Jabber, un protocole ouvert basé sur XML utilisé pour la messagerie instantanée, qui permet aux utilisateurs de différents serveurs de communiquer entre eux.

Dans l'article [50], Dian Neipa et al. proposent une solution pour sécuriser le partage des dossiers médicaux électroniques grâce au système IBE-ECC-ECIES. L'Identity-Based Encryption (IBE) simplifie la gestion des clés en utilisant l'identité de l'utilisateur (comme un e-mail ou une adresse IP) pour créer la clé publique, ce qui évite d'avoir

à gérer des certificats compliqués. Leur méthode utilise aussi ECC pour sécuriser les échanges de clés (ECDH), la signature numérique (ECDSA), et pour chiffrer les informations de manière hybride avec ECIES. Les tests montrent que le système est rapide : la création des clés prend 0,6 seconde, et le chiffrement ou déchiffrement seulement 0,2 seconde. Il résiste bien aux attaques comme les interceptions. Ce système assure un accès sécurisé et contrôlé aux dossiers, garde une trace des accès, et fonctionne aussi bien sur mobile. Les auteurs envisagent d'adapter cette solution aux objets connectés médicaux et d'améliorer ses performances pour les infrastructures hospitalières.

Dans l'article [51], Xia Lin présente une nouvelle méthode pour mieux protéger les e-mails. Il explique que le protocole SMTP, qui sert à envoyer les e-mails, ne chiffre pas les messages par défaut. Cela signifie que les messages peuvent être interceptés et lus par des personnes mal intentionnées pendant leur envoi.Pour résoudre ce problème, l'auteur propose une méthode de chiffrement hybride qui combine deux techniques : le 3-DES, utilisé pour chiffrer rapidement le contenu des e-mails, et les courbes elliptiques (ECC), qui servent à échanger les clés de façon sécurisée et à ajouter une signature numérique (ECDSA) pour garantir l'authenticité du message.Cette méthode offre un bon niveau de sécurité avec des performances optimisées grâce à l'usage de clés plus courtes que RSA. L'auteur recommande d'intégrer cette solution dans des standards comme S/MIME, et d'explorer des améliorations matérielles pour faire face aux menaces futures, comme l'informatique quantique.

Dans l'article [52], D. Jagadessh *et al.* proposent une solution au problème de la transmission sécurisée de données sensibles, comme les images, face aux menaces d'interception, de modification ou d'accès non autorisé.Pour y répondre, leur méthode combine quatre techniques : la cryptographie à courbes elliptiques (ECC) pour assurer la confidentialité, le hachage SHA-3 pour vérifier l'intégrité, la compression de Huffman pour réduire la taille des données, et la stéganographie pour dissimuler les données chiffrées dans une image. Les résultats expérimentaux sont prometteurs : réduction de la taille des données de 15 % à 25 %, détection d'altérations avec une précision de 99,9 %, aucune distorsion visible sur les images, et un taux de récupération des données de 100 %, le tout avec une faible charge de calcul.En perspective, cette approche pourrait être adaptée aux systèmes temps réel, intégrée à des plateformes cloud ou blockchain, et renforcée par des techniques d'intelligence artificielle
Le tableau 2.7 résume les principales caractéristiques de ces approches :

| Référence | Contexte/Objectif | Techniques utilisées | Avantages principaux | Perspectives | Limites |
|---|---|---|---|---|---|
| Alhaj (2024) [44] | Sécurisation efficace des données, alternative à RSA | ECC (ECDSA, ECDH), comparaison avec RSA, AES, ChaCha20 | Clés plus courtes, efficacité énergétique, adapté aux IoT | Comparaison théorique, pas de mise en œuvre spécifique | Analyse principalement théorique, pas de test réel, pas de prise en compte post-quantique |
| Gayathri et al. (2017) [45] | Solution hybride de chiffrement rapide et sécurisé | AES-192 + ECC-256 | Résistance accrue aux attaques, meilleure sécurité hybride | Vers AES-256/ECC-384 pour plus de robustesse future | des tests limités surtout en temps réels |
| Mitra et al. (2010) [46] | Protocole de chiffrement email basé sur ECC | ECC sur $GF(2^m), signaturesnumriques$ | Clés courtes, rapidité, meilleure résistance aux attaques | Calculs complexes, gestion des clés à optimiser | Calculs lourds, gestion des clés perfectible (incomplète) , pas de validation terrain() en temps réel |
| Neelima et al. (2025) [47] | Amélioration du chiffrement ElGamal sur ECC | Polynômes de Chebyshev, signatures numériques, rotations binaires | Texte chiffré plus compact, résistance aux attaques ECDLP | Complexité algorithmique, implémentation pratique à valider | Complexité mathématique, pas d'implémentation à grande échelle |
| Murthy et al. (2024) [48] | Améliorer chiffrement email face aux limites de RSA | ECC, S/MIME, PGP, TensorFlow, NumPy | Chiffrement rapide (0.05 ms), compatible systèmes existants | Intégration dans divers environnements à approfondir | Tests et évaluation limités |

| Référence | Contexte/Objectif | Techniques utilisées | Avantages principaux | Perspectives | Limites |
|---|---|---|---|---|---|
| Yang et al. (2008) [49] | Sécuriser messagerie instantanée | SIMPP (ECDSA, ECDH, AES-128, HMAC-SHA-256), compatible XMPP | clés courtes, adapté mobiles | Extension à d'autres protocoles possible | Architecture complexe, sécurité non formellement prouvée, pas résistante au quantique |
| Neipa et al. (2019) [50] | Sécuriser partage dossiers médicaux via IBE-ECC-ECIES | IBE, ECC (ECDH, ECDSA, ECIES) | Gestion simplifiée des clés, rapide, sécurisé | Optimisation pour objets connectés médicaux envisagée | Dépendance au PKG (par des autorités ), révocation de clé difficile, coût élevé sur objets connectés |
| Lin (2016) [51] | Chiffrement hybride pour sécuriser email SMTP | 3-DES + ECC (ECDSA), signature numérique | Confidentialité, intégrité, non-répudiation, clé courte | Adaptation aux standards S/MIME, futur matériel quantique | Usage de 3-DES obsolète, absence de validation réelle, pas de sécurité post-quantique |
| Jagadessh et al. (2025) [52] | Transmission sécurisée d'images | ECC, SHA-3, compression Huffman, stéganographie | Réduction taille, haute intégrité, aucune distorsion visuelle | Adaptation à temps réel, IA pour détection d'anomalies | pas testé en temps réel |

TABLE 2.2 – Analyse comparative des solutions de sécurisation de messagerie par ECC.

**Synthèse et discussion générale**

Notre études aux travaux récents met en évidence la flexibilité et l'efficacité des solutions basées sur la cryptographie à courbes elliptiques (ECC) pour la sécurisation des systèmes de messagerie. Ces solutions se démarquent par leur capacité à combiner performance, innovation et adaptabilité.

Parmi les approches les plus remarquables, on trouve les modèles hybrides associant à des algorithmes symétriques comme AES, les améliorations mathématiques à l'aide de polynômes spéciaux (comme Chebyshev), ainsi que l'intégration de nouvelles technologies telles que la blockchain. Ces contributions permettent des avancées concrètes :

- **Renforcement de la sécurité :** Cela signifie que les solutions utilisent ECC offrent un niveau de protection plus élevé contre les attaques (Grâce à la robustesse des primitives utilisées comme ECDLP, AES, et SHA-256).
- **Réduction du temps de traitement :** Cela fait référence à l'efficacité de calcul des algorithmes modernes basées sur l'ECC ( clés plus courtes et optimisations algorithmiques).
- **Adaptation aux besoins des applications modernes :** Cela fait référence à la souplesse et à la compatibilité de ces solutions avec des technologiques actuelles (PKI moderne, WhatsApp, TLS, Bitcoin ).

Néanmoins,ces méthodes restent confrontées à certains défis, notamment la complexité des calculs et une vulnérabilité potentielle face aux évolutions futures de l'informatique quantique.

## 2.8 Conclusion

Ce chapitre, nous a permis de mettre en évidence les bases théoriques et pratiques des courbes elliptiques, et leurs impacts sur la cryptographie. Cependant, bien que plusieurs objectifs aient été atteints, certaines limites persistent, notamment face aux évolutions rapides des technologies. Ces constats nous ont conduit à explorer de nouvelles pistes, en particulier l'hybridation entre la cryptographie à courbes elliptiques (ECC) et les algorithmes post-quantiques.

Chapitre 3

# PROPOSITION D'UN SYSTÈME CRYPTOGRAPHIQUE HYBRIDE

## 3.1 Introduction

Á l'émergence des ordinateurs quantiques, les systèmes de cryptographie actuels comme l'ECC deviennent vulnérables aux algorithmes quantiques tels que Shor, rendant nécessaire le développement de nouvelles méthodes de chiffrement résistantes à ces menaces pour garantir la sécurité des messageries électroniques.

Ce chapitre présente une solution cryptographique hybride nommée (EKDA) qui combine la cryptographie moderne et le post-quantique, accompagnée d'une analyse comparative avec d'autres solutions cryptographiques existantes.

## 3.2 Conception d'un système de chiffrement hybride

Notre solution repose sur une architecture hybride cryptographique combinant des primitives classiques et post-quantiques pour assurer une sécurité renforcée et durable. Son principe fondamental repose sur une double protection : même si l'une des deux méthodes cryptographiques est compromise, l'autre maintient la sécurité du système. L'architecture s'articule autour de deux phases : la cryptographie asymétrique établit un canal sécurisé et échange les clés de session sur des canaux non fiables, puis la cryptographie symétrique prend le relais pour chiffrer efficacement les données. Cette approche concilie robustesse lors de l'initialisation et performance lors des transferts, créant une solution adaptée aux besoins actuels tout en résistant aux menaces quantiques future.

## 3.3 Présentation de la solution proposée

L'architecture proposée intègre les composants cryptographiques suivants, chacun remplit un rôle spécifique dans le système de sécurité :

### 3.3.1 Application Cryptographie sur Courbes Elliptiques (ECC)

**Spécifications**

- **Courbe utilisée : SECP384R1** : (Standards for Efficient Cryptography Prime), avec une clé de taille 384 bits. (voir le tableau 2.1).
- **Équation de la courbe :** $y^2 = x^3 - 3x + b \pmod{p}$.

**Génération des clés**

- **Clé privée (d) :** Entier aléatoire dans l'intervalle $[1, n-1]$.
- **Clé publique (Q) :** Point sur la courbe tel que $Q = d \cdot G$, où $G$ est le générateur.

**Rôle de ECC dans le système**

L'ECC fournit la composante classique de l'échange de clés via le protocole ECDH. Cette primitive éprouvée garantit la compatibilité avec les infrastructures existantes.

### 3.3.2 Kyber

C'est un mécanisme d'encapsulation de clé (KEM) asymétrique post-quantique fondé sur la cryptographie à base des réseaux modulaires. Développé pour résister aux attaques des ordinateurs quantiques [53].

**Spécifications**

- **Clé privée :** 32 octets (représentant un vecteur de polynômes secrets).
  Elle doit rester secrète et connue uniquement par le destinataire, utiliser pour la décapsulation.
- **Clé publique :** 800 octets (matrice de polynômes publics).
  Elle est utilisée dans l'encapsulation pour produire le seceret partagé.
- **Texte chiffré :** 768 octets.
- **Secret partagé :** 32 octets (256 bits).

**Rôle dans le système**

Kyber est un outil cryptographique permet à deux personnes d'échanger un secret en toute sécurité, même face à des ordinateurs très puissants comme les ordinateurs quantiques. Ce secret sert ensuite à créer des clés qui protègent les échanges de données. Par exemple, dans une application de messagerie, Kyber permet de générer une clé secrète partagée entre l'expéditeur et le destinataire pour chiffrer les messages. Ainsi, même si un pirate intercepte le message, il ne pourra pas le lire. Car, il ne possède pas cette clé secrète garantissant la sécurité des communications dans le futur.

**Algorithmes du fonctionnement de Kyber**

Les algorithmes suivants détaillent les trois étapes essentielles : génération de clés, encapsulation, et décapsulation.

- **Algorithme 3 : Génération de la paire de clés**

  Cette étape permet de générer la clé publique (à partager) et la clé privée (à garder secrète). Elle repose sur la génération de bruit aléatoire et la construction d'une matrice pseudo-aléatoire.

**Algorithm 3** Génération de la paire de clés Kyber

**Entrée :** Paramètres de sécurité $(k, q, n, \eta)$
**Sorties :** Clé publique $pk$, clé privée $sk$
Générer matrice aléatoire $A \in R_q^{k \times k}$ à partir d'une graine aléatoire
Générer vecteur secret $s \in R_q^k$ selon distribution centrée
Générer bruit $e \in R_q^k$
Calculer $t \leftarrow A \cdot s + e \mod q$
**Clé publique :** $pk \leftarrow (A, t)$
**Clé privée :** $sk \leftarrow s$
**Retourner** $(pk, sk)$

- **Algorithme 4 : Encapsulation de la clé partagée**

  Cette étape est utilisée par l'expéditeur pour encapsuler une clé secrète partagée à partir de la clé publique du destinataire. Elle produit une capsule chiffrée qui peut être envoyée publiquement.

**Algorithm 4** Encapsulation d'une clé avec Kyber

**Entrée :** Clé publique $pk = (A, t)$
**Sorties :** Capsule chiffrée $ct = (u, v)$, clé partagée $K$
Générer message aléatoire $m \in \{0, 1\}^n$
Calculer hash : $m' \leftarrow SHA3\text{-}256(m)$
Graine $seed \leftarrow SHA3\text{-}256(m' \parallel pk)$
Générer vecteurs d'erreur $r, e_1, e_2$ à partir de $seed$
Calculer $u \leftarrow A^T \cdot r + e_1 \mod q$
Calculer $v \leftarrow t^T \cdot r + e_2 + Encode(m) \mod q$
Calculer $K \leftarrow SHA3\text{-}256(u \parallel v \parallel m)$
**Retourner** $(ct = (u, v), K)$

- **Algorithme 5 : Décapsulation de la clé partagée**

Cette étape est réalisée par le destinataire. Elle permet de récupérer la clé partagée à partir de la capsule reçue, en vérifiant l'intégrité du message.

**Algorithm 5** Décapsulation d'une clé avec Kyber

**Entrées :** Capsule $ct = (u, v)$, clé privée $sk = s$, clé publique $pk = (A, t)$
**Sortie :** Clé partagée $K$
Calculer $m' \leftarrow Decode(v - u^T \cdot s \mod q)$
Hash $m'' \leftarrow SHA3\text{-}256(m')$
Recalculer $seed \leftarrow SHA3\text{-}256(m'' \parallel pk)$
Générer $r', e'_1, e'_2$ à partir de $seed$
Recalculer $u' \leftarrow A^T \cdot r' + e'_1 \mod q$
Recalculer $v' \leftarrow t^T \cdot r' + e'_2 + Encode(m') \mod q$ **if** $(u, v) = (u', v')$ **then**

    $K \leftarrow SHA3\text{-}256(u \parallel v \parallel m')$ **else**

    $K \leftarrow SHA3\text{-}256(z)$ ▷ $z$ : valeur alternative sécurisée

**Retourner** $K$

### 3.3.3 Dilithium

C'est un algorithme de signature numérique asymetrique conçu pour résister aux attaques des ordinateurs quantiques. Il repose sur la technique « Fiat-Shamir with Aborts » appliquée aux réseaux euclidiens. Sa sécurité est fondée sur le problème M-LWE (Module Learning With Errors), qui reste difficile à résoudre même pour les ordinateurs quantiques [54].

#### Spécifications

- **Clé privée :** 2500 octets (contient des matrices et vecteurs secrets permettant de générer une signature).
- **Clé publique :** 1500 octets (contient des matrices et vecteurs publics permettant de vérifier une signature).
- **Signature :** 3300 octets (preuve mathématique jointe au message pour garantir son authenticité).

#### Rôle dans un système de messagerie

Dans une application de messagerie sécurisée :

- L'expéditeur utilise sa **clé privée** pour signer numériquement le message. Cette signature permet de prouver l'origine du message.
- Le destinataire utilise la **clé publique** de l'expéditeur pour vérifier la validité de la signature. Si la vérification réussit, cela signifie que le message provient bien de l'expéditeur et n'a pas été altéré.

Ainsi, même face aux menaces futures et aux ordinateurs quantiques, Dilithium assure l'**authenticité** et l'**intégrité** des messages échangés.

**Algorithmes du fonctionnement de Dilithium**

Les algorithmes suivants détaillent les trois étapes essentielles : génération de clés, signature, et vérification.

- **Algorithme 6 : Génération de la paire de clés**
  Cette étape génère une clé publique (utilisée pour la vérification) et une clé privée (utilisée pour signer). La sécurité repose sur la difficulté du problème Module-LWE.

**Algorithm 6** Génération de la paire de clés Dilithium

1: **Entrée :** Paramètres de sécurité $(k, q, \eta, \gamma_1, \gamma_2)$
2: **Sorties :** Clé publique $pk$, clé privée $sk$
3: Générer matrice aléatoire $A \in R_q^{k \times l}$ à partir d'une graine
4: Échantillonner vecteurs secrets $s_1 \in R_q^l$, $s_2 \in R_q^k$
5: Calculer $t \leftarrow A \cdot s_1 + s_2 \mod q$
6: Décomposer $t = (t_1, t_0)$ où $t_1$ est public, $t_0$ est secret
7: **Clé publique :** $pk \leftarrow (A, t_1)$
8: **Clé privée :** $sk \leftarrow (s_1, s_2, t_0)$
9: **Retourner** $(pk, sk)$

- **Algorithme 7 : Signature numérique**
  L'expéditeur utilise sa clé privée pour signer un message. La signature repose sur un engagement, un challenge, et une réponse, comme dans les schémas de Fiat-Shamir à trois tours.

**Algorithm 7** Signature d'un message avec Dilithium

**Entrées :** Message $M$, clé privée $sk = (s_1, s_2, t_0)$
**Sortie :** Signature $\sigma = (z, h)$
Générer aléatoirement $y \in R_q^l$
Calculer $w \leftarrow A \cdot y \mod q$
Compresser $w$ pour obtenir $w_1$
Calculer challenge $c \leftarrow H(w_1, M)$
Calculer réponse $z \leftarrow y + c \cdot s_1$
Calculer masque $h$ pour cacher certaines composantes
Vérifier que $z$ respecte les bornes fixées par $\gamma_1$
**Retourner** $\sigma = (z, h)$

- **Algorithme 8 : Vérification de signature**
  Le destinataire vérifie l'authenticité du message en reconstruisant l'engagement et le challenge, à partir de la clé publique et de la signature reçue.

**Algorithm 8** Vérification d'une signature avec Dilithium

**Entrées :** Message $M$, signature $\sigma = (z, h)$, clé publique $pk = (A, t_1)$
**Sortie :** Validité $v \in \{\texttt{true}, \texttt{false}\}$
Vérifier que $z$ respecte les bornes admissibles
Calculer $w' \leftarrow A \cdot z - c \cdot t_1 \mod q$
Appliquer masque $h$ pour obtenir $w_1'$
Calculer $c' \leftarrow H(w_1', M)$ **if** $c' = c$ **then**
    **Retourner** `true` **else**
    **Retourner** `false`

### 3.3.4 HKDF (HMAC-based Key Derivation Function)

Une fonction de dérivation clé, défini dans RFC 5869. Elle repose sur l'algorithme HMAC (généralement SHA-256), et suit un paradigme "Extract-then-Expend" en deux étapes [55] :

- **Extract :** Phase d'extraction d'un aléa.
- **Expand :** Phase d'expansion qui génère une ou plusieurs clés cryptographiques de longueur adaptés à divers usages tels que le chiffrement et l'authentification.

**Rôle dans le système**

Dans un système de messagerie sécurisé, HKDF permet de dériver une clé de session unique et sécurisée à partir des secrets établis par ECDH et Kyber. Cette clé de session est ensuite utilisée pour chiffrer les messages échangés entre l'expéditeur et le destinataire, garantissant la confidentialité et l'intégrité des communications, même face à des attaquants équipés d'ordinateurs quantiques.

### 3.3.5 AES-GCM

AES-GCM est un algorithme de chiffrement symétrique qui assure à la fois la confidentialité et l'authenticité des données.

**Spécifications**

- **Algorithme** : AES-256 en mode GCM (Galois/Counter Mode).
- **Taille de clé** : 256 bits (clé de chiffrement dérivée via HKDF).
- **Nonce** : 96 bits (12 octets)
- **Tag d'authentification** : 128 bits.

**Explication**

**AES-GCM** est un algorithme de chiffrement symétrique offrant à la fois la confidentialité (via le chiffrement) et l'intégrité des données (via le tag d'authentification fournit par le mode GCM).En utilisant des clés de 256 bits dérivées par HKDF qui combine les secrets issus des algorithmes ECDH et Kyber.

Le mode **GCM** a été choisi car il assure simultanément la confidentialité et l'intégrité des données échangées, ce qui est essentiel pour une messagerie sécurisée. De plus, il est rapide, efficace et adapté aux environnements modernes, notamment grâce à sa capacité de traitement parallèle et sa compatibilité avec les clés dérivées via HKDF.

Le **nonce** est une variable garantit que le même message chiffré deux fois donnera des sorties différentes, renforçant la sécurité contre les attaques par relecture ou duplication.

Le **tag d'authentification** de 128 bits permet au destinataire de vérifier l'intégrité et l'authenticité des données reçues. Si le tag ne correspond pas à celui attendu, le message est considéré comme falsifié ou corrompu.

**Rôle dans le système**

Dans le cadre d'une messagerie sécurisée, AES-GCM assure que les messages échangés

restent confidentiels et ne peuvent être ni modifiés ni falsifiés par un attaquant. Même si le message est intercepté, il ne pourra être déchiffré ni modifié sans la clé correcte, garantissant la sécurité des échanges dans le présent et le futur.

### 3.3.6 Pseudo-code de la proposition cryptographique : ECC+Kyber +Dilithium+AES_GCM (EKDA)

Le pseudo-code 9 présente une suite d'algorithmes constituant notre proposition EKDA :

**Algorithm 9** Algorithme Principal : EKDA

**Require:** Message $M$ ou fichier $F$ à sécuriser.

**Ensure:** Résultat de la vérification d'intégrité et contenu déchiffré.

**Phase 1 : Sécurisation**

$(C, \sigma, pk_{\text{ECC}}, pk_{\text{Kyber}}, pk_{\text{Dilithium}}, ct_{\text{Kyber}}, nonce) \leftarrow$ **Algorithme 10.**

**Phase 2 : Vérification**

$(M' ou F', \text{valid}) \leftarrow$ **Algorithme 11**.

**if** *valid* = `true` **then**

**return** "Sécurisation réussie : contenu authentique et intègre".

**else**

**return** "Échec de sécurisation : contenu compromis ou altéré".

- **Chiffrement et signature Hybride**

  Illustré dans l'algorithme 10 ci-dessous :

**Algorithm 10** Étape 1 : Chiffrement et Signature Hybride Post-Quantique

**Entrées :** Message $M$ ou fichier $F$ à sécuriser.

**Sorties :** Données chiffrées $C$, signature $\sigma$, clés publiques, capsule Kyber $ct_{kyber}$, nonce.

**Génération des clés cryptographiques**

Générer paire de clés ECC : $(sk_{ecc}, pk_{ecc}) \leftarrow ECC.KeyGen()$.

Générer paire de clés Kyber : $(sk_{kyber}, pk_{kyber}) \leftarrow Kyber.KeyGen()$.

Générer paire de clés Dilithium : $(sk_{dil}, pk_{dil}) \leftarrow Dilithium.KeyGen()$.

**Établissement de la clé hybride**

Encapsulation Kyber : $(ct_{kyber}, K_{kyber}) \leftarrow Kyber.Encaps(pk_{kyber})$.

Calculer secret ECDH : $K_{ecdh} \leftarrow ECDH(sk_{ecc}, pk_{ecc})$.

Combiner les secrets : $K_{comb} \leftarrow SHA3\text{-}256(K_{ecdh}) \parallel K_{kyber}$.

Dériver clé AES : $K_{aes} \leftarrow HKDF(K_{comb}, salt = \emptyset, info =$ "hybrid").

**Chiffrement du contenu**

Générer nonce aléatoire : $nonce \leftarrow \{0,1\}^{96}$.

**if** *(entrée est un message M)* **then**

$C \leftarrow AES\text{-}GCM.Encrypt(K_{aes}, nonce, M)$.

**else**

(entrée est un fichier $F$)

Découper $F$ en blocs de 1 Mo : $\{B_1, B_2, \ldots, B_n\}$

**for** $i = 1$ ***to*** $n$ **do**

$C_i \leftarrow AES\text{-}GCM.Encrypt(K_{aes}, nonce + i, B_i)$.

$C \leftarrow \{C_1, C_2, \ldots, C_n\}$.

**Signature numérique**

Calculer empreinte : $hash \leftarrow SHA3\text{-}256(M$ ou $F)$.

Signer avec Dilithium : $\sigma \leftarrow Dilithium.Sign(sk_{dil}, hash)$.

**Retourner** $(C, \sigma, pk_{ecc}, pk_{kyber}, pk_{dil}, ct_{kyber}, nonce)$.

- **Déchiffrement et vérification de signature**

  Illustré dans l'algorithme 11 ci-dessous :

**Algorithm 11** Étape 2 : Déchiffrement et Vérification de Signature

**Entrées :** $C$, $\sigma$, $sk_{ecc}$, $sk_{kyber}$, $pk_{ecc}$, $pk_{dil}$, $ct_{kyber}$, $nonce$.

**Sorties :** Contenu déchiffré $M'$ ou $F'$, résultat de vérification $valid$.

**Reconstitution de la clé de déchiffrement**

Décapsuler Kyber : $K'_{kyber} \leftarrow Kyber.Decaps(sk_{kyber}, ct_{kyber})$.

Recalculer secret ECDH : $K'_{ecdh} \leftarrow ECDH(sk_{ecc}, pk_{ecc})$.

Recombiner les secrets : $K'_{comb} \leftarrow SHA3\text{-}256(K'_{ecdh}) \parallel K'_{kyber}$.

Redériver clé AES : $K'_{aes} \leftarrow HKDF(K'_{comb}, salt = \emptyset, info =$ "hybrid").

**Déchiffrement du contenu**

**if** $C$ *est un message chiffré unique* **then**

$M' \leftarrow AES\text{-}GCM.Decrypt(K'_{aes}, nonce, C)$.

**else**

$C = \{C_1, C_2, \ldots, C_n\}$. est un ensemble de blocs.

**for** $i = 1$ ***to*** $n$ **do**

$B'_i \leftarrow AES\text{-}GCM.Decrypt(K'_{aes}, nonce + i, C_i)$.

Reconstituer fichier : $F' \leftarrow B'_1 \parallel B'_2 \parallel \ldots \parallel B'_n$.

**Vérification de la signature**

Calculer empreinte du contenu déchiffré : $hash' \leftarrow SHA3\text{-}256(M'$ ou $F')$.

Vérifier signature : $valid \leftarrow Dilithium.Verify(pk_{dil}, hash', \sigma)$.

**if** $valid =$ `true` **then**

**Afficher** " Signature valide - Intégrité confirmée".

**Autoriser** l'utilisation du contenu déchiffré. **else**

**Afficher** " Signature invalide - Intégrité compromise".

**Rejeter** le contenu déchiffré

$M' \leftarrow \emptyset$ ou $F' \leftarrow \emptyset$.

**Mesures de performance**

Mesurer et afficher : temps d'exécution, utilisation CPU, utilisation RAM.

**Retourner** $(M'$ ou $F', valid)$.

## 3.4 Analyse Comparative des Performances d'Algorithmes Cryptographiques

L'étude propose une analyse comparative approfondie des performances de quatre configurations cryptographiques distinctes : une solution hybride post-quantique décrite dans l'algorithme 9, comparée à trois schémas classiques, à savoir : **ECC+ECDSA+AES_CCM** (solution elliptique conventionnelle), **RSA** (cryptographie asymétrique ) et **DES-CBC** (méthode symétrique).

L'évaluation porte sur cinq opérations cryptographiques fondamentales, exécutées sur un large éventail de tailles de données allant de 10 Ko à 100 Mo. Les performances sont mesurées à l'aide de trois indicateurs clés : le **temps d'exécution**, la **consommation du processeur (CPU %)**, et l'**utilisation de la mémoire vive (RAM)**.

## 3.5 Méthodologie

- **Scénario A** : décrit une solution hybride post-quantique **EKDA** (voir l'algorithme 9).
- **Scénario B** : décrit l'algorithme **DES-CBC** (Data Encryption Standard en mode Cipher Block Chaining) (voir l'algorithme 12).
- **Scénario C** : décrit l'algorithme hybride **ECC+ECDSA+AES_CCM** (voir l'algorithme 14).
- **Scénario D** : décrit l'algorithme asymétrique **RSA** (Rivest-Shamir-Adleman) (voir l'algorithme 1).

### 3.5.1 Pseudo-code de l'algorithme DES-CBC

L'algorithme 12 décrit le processus de chiffrement DES en mode CBC :

**Algorithm 12** Algorithme DES_CBC

1: **Entrée :** Message clair $M$ divisé en blocs $M_1, M_2, \ldots, M_n$ de 64 bits, clé $K$, vecteur d'initialisation $IV$.
2: **Sortie :** Texte chiffré $C$ divisé en blocs $C_1, C_2, \ldots, C_n$.
3: $C_0 \leftarrow IV$. **for** $i \leftarrow 1$ ***to*** $n$ **do**
4: $T \leftarrow M_i \oplus C_{i-1}$. ▷ XOR du bloc clair avec le bloc chiffré précédent.
6: $C_i \leftarrow DES_{enc}(K, T)$. ▷ Chiffrement du bloc avec DES.
7: **Retourner** $C = C_1 \| C_2 \| \ldots \| C_n$.

### 3.5.2 Pseudo-code du déchiffrement DES-CBC

L'algorithme 13 décrit le processus de déchiffrement DES en mode CBC :

**Algorithm 13** Déchiffrement DES_CBC

1: **Entrée :** Texte chiffré $C$ divisé en blocs $C_1, C_2, \ldots, C_n$ de 64 bits, clé $K$, vecteur d'initialisation $IV$.
2: **Sortie :** Message clair $M$ divisé en blocs $M_1, M_2, \ldots, M_n$.
3: $C_0 \leftarrow IV$.
4: **for** $i \leftarrow 1$ ***to*** $n$ **do**
5: $T \leftarrow DES_{dec}(K, C_i)$. ▷ Déchiffrement du bloc avec DES
6: $M_i \leftarrow T \oplus C_{i-1}$. ▷ XOR avec le bloc chiffré précédent.
7: **Retourner** $M = M_1 \| M_2 \| \ldots \| M_n$.

### 3.5.3 Pseudo-code de l'algorithme hybride ECC-AES_CCM

L'algorithme 14 décrit l'approche hybride (ECC, ECDSA , AES-CCM) :

**Algorithm 14** Algorithme Hybride ECC-AES_CCM

**Entrée :** $p$ premier, courbe elliptique sécurisée (ex : SECP384R1).
**Sortie :** Texte chiffré et signature $(c, \sigma)$.
**Phase de génération des clés ECC**
Choisir $d \in [1, n-1]$ aléatoirement. ▷ Clé privée.
Calculer $Q = d \times G$. ▷ Clé publique (point sur la courbe).
Stocker $(d, Q)$.
**Phase d'échange ECDH + Dérivation de la clé AES**
Calculer $S = d_A \times Q_B$. ▷ Secret partagé via ECDH.
$k \leftarrow \text{HKDF}(S, \text{info} = \text{"AES-KEY"})$. ▷ Dérivation d'une clé symétrique de 256 bits.
**Phase de chiffrement AES (mode CCM)**
$nonce \leftarrow$ Aléatoire(12 octets).
$c \leftarrow \text{AES-CCM-Encrypt}(k, nonce, m)$.
Message chiffré : $(nonce \parallel c)$.
**Phase de signature ECDSA**
$h \leftarrow \text{SHA-256}(m)$.
Choisir $k \in [1, n-1]$ aléatoirement.
Calculer $(r, s) = \text{SignatureECDSA}(d, h, k)$.
Signature : $(r, s)$.
**Workflow complet**.
$(d_A, Q_A) \leftarrow \text{GénérerClés}()$.
$(d_B, Q_B) \leftarrow \text{GénérerClés}()$.
$k_A \leftarrow \text{DériverCléSymétrique}(d_A, Q_B)$.
$k_B \leftarrow \text{DériverCléSymétrique}(d_B, Q_A)$.
$c \leftarrow \text{Chiffrer}(k_A, m)$.
$\sigma \leftarrow \text{Signer}(d_A, m)$.
**Retourner** $(c, \sigma)$.

### 3.5.4 Pseudo-code de l'algorithme RSA

L'algorithme 1 décris da le premier chapitre, présente le processus de chiffrement et de déchiffrement par la méthode cryptographie RSA.

### 3.5.5 Calcul des mesures de performances

Afin de garantir une évaluation rigoureuse des performances cryptographiques, cette étude s'appuie sur une méthodologie expérimentale clairement définie. Les paramètres suivants ont été pris en compte pour les mesures.

#### 3.5.5.1 Variables Indépendantes

Les facteurs contrôler dans une expérimentation utilisés pour analyser leurs impacts sur les résultats mesuré, à savoir :

- **Taille des jeux de données** : Dataset incluent des fichiers générés manuellement **(.txt et .py)** et dautres importés depuis des sources externes (**.csv** dans le drive) avec des tailles varient entre **(10Ko,100Mo)**.
- **Type d'opération cryptographique** : Génération de clés, chiffrement, déchiffrement, signature numérique, vérification de signature.

#### 3.5.5.2 Variables Dépendantes

Résultats mesurés qui dépendent des variables indépendants. Ils sont utiliser pour quantifier les performances selon les conditions testées .

- **Temps d'exécution** : mesuré en secondes avec une précision milliseconde pour chaque opération cryptographique.

#### 3.5.5.3 Performance en termes de ressources

L'évaluation des performances incluent également des indicateurs essentiels et basiques tels que l'utilisation du processeur (CPU), et la consommation mémoire (RAM), afin de mesurer l'efficacité globale du système.

- **Utilisation du processeur (CPU)** : exprimée en pourcentage.
- **Consommation mémoire (RAM)** : mesurée en kilooctets (Ko).

### 3.5.6 Environnement de développement

Pour assurer la répétabilité des résultats obtenus, une description claire et détaillée de l'environnement de test est nécessaire.

- **Plateforme** : Google Colab (environnement virtuel Linux).

- **Langage** : Python 3.11.


FIGURE 3.1 – Python et Google colab [56].

- **Bibliothèques**
  - `cryptography` : implémentation des primitives cryptographiques (ECC, AES-GCM, HKDF).
  - `pycryptodome` : utilisation de l'algorithme de hachage SHA3-256 et génération de valeurs aléatoires sécurisées.
  - `time` : chronométrage des opérations d'exécution.
  - `psutil` : surveillance de l'utilisation des ressources système (CPU, mémoire).
  - `tracemalloc` : suivi de l'allocation mémoire durant l'exécution du programme.
  - `pandas` : organisation et analyse des données collectées (par exemple, résultats de performance).
  - `tqdm` : affichage d'une barre de progression pour visualiser l'avancement des tâches.
  - `zipfile` : compression et décompression des fichiers (par exemple, pour archiver les résultats).
  - `google.colab.files` : gestion des fichiers sous Google Colab (import/export).
  - `matplotlib.pyplot` : création de graphiques pour visualiser les résultats et les performances.
  - `numpy` : manipulation de tableaux numériques et réalisation de calculs mathématiques avancés.

## 3.6 Résultats et Analyse

Cette section présente les performances de génération de clé, de chiffrement déchiffrement, ainsi que la signature et de sa vérification.

### 3.6.1 Performance de Génération de Clés

La génération de paires de clés constitue une opération fondamentale dans tout système cryptographique.

La Figure 3.2 présente les résultats comparatifs pour les quatre algorithmes étudiés :

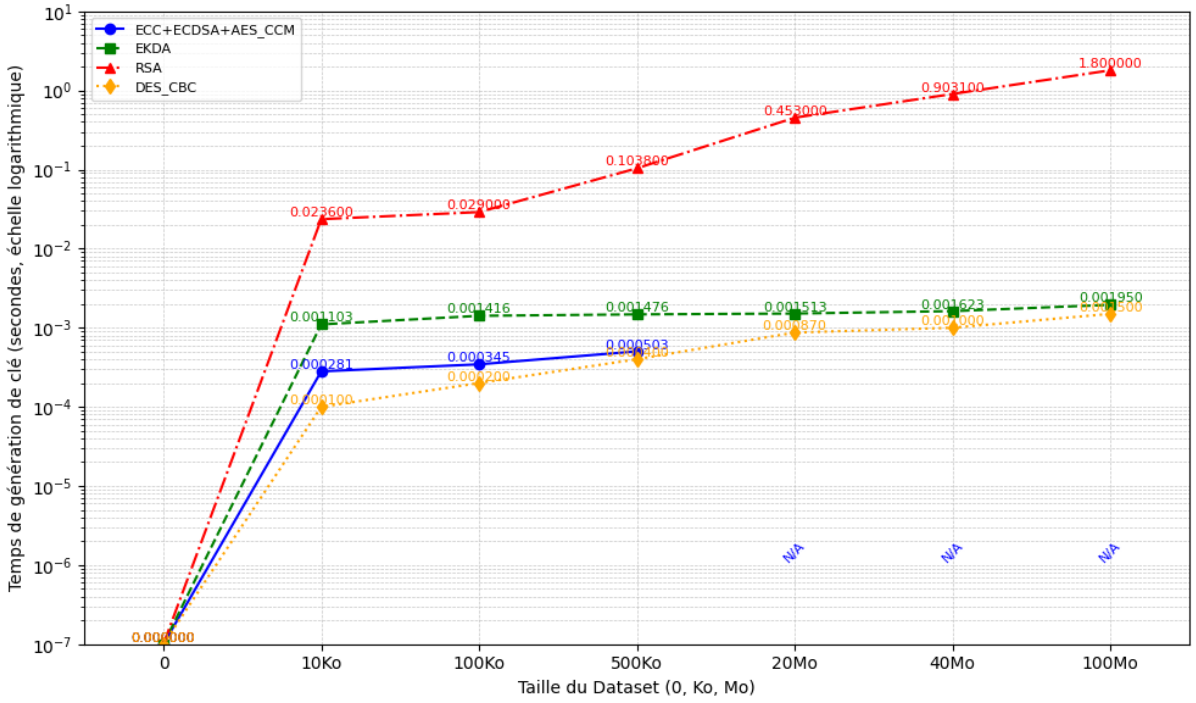


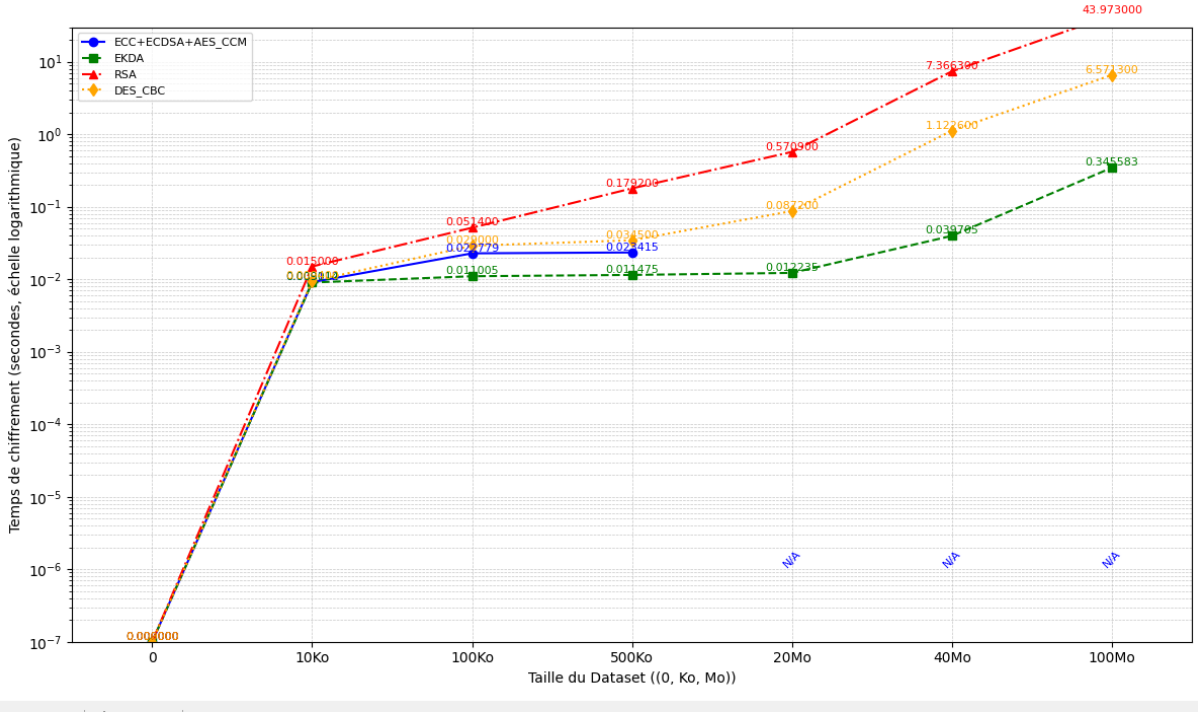

FIGURE 3.2 – Performances comparées des temps de génération de paires de clés pour les quatre algorithmes.

L'analyse des temps de génération 3.2 des clés révèle des différences significatives entre les configurations testées. La configuration **DES_CBC** affiche les temps de génération les plus faibles, démontrant ainsi une grande rapidité pour cette opération.

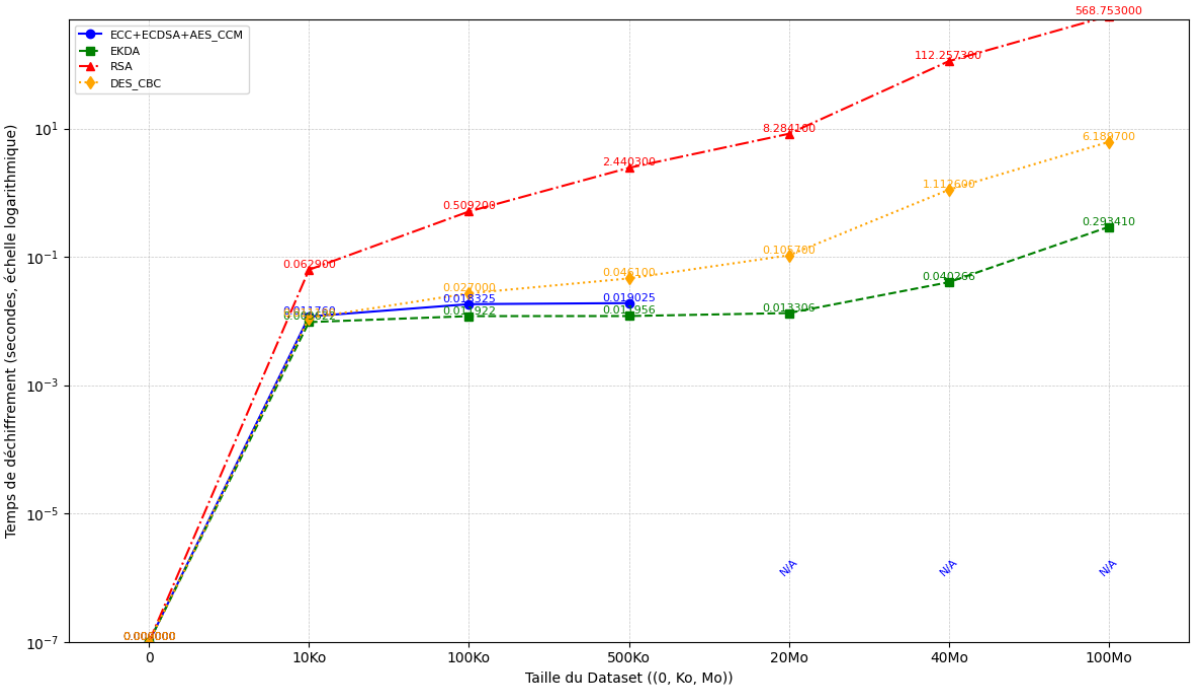
La combinaison **ECC+ECDSA+AES_CCM** présente également de bonnes performances, bien qu'elle rencontre certaines limitations pour les fichiers volumineux (supérieurs à 1 Mo) en raison des contraintes limités au mode CCM. Cependant, notre solution hybride proposée, intégrant **ECC+Kyber+Dilithium+AES_GCM** , elle se caractérise par des temps de génération modérés mais d'une grande stabilité, quelle que soit les fichiers de taille variable. Enfin, l'algorithme **RSA** demeure le plus lent parmi les solutions comparées, en raison de la complexité liée à la génération de grands nombres premiers nécessaires à sa clé.

### 3.6.2 Performances de Chiffrement et Déchiffrement

La figure suivante illustre le temps de chiffrement et de déchiffrement en fonction de taille de données (Dataset) :




(a) Temps de chiffrement des algorithmes en fonction de la taille des données.




(b) Temps de déchiffrement des algorithmes en fonction de la taille des données.

FIGURE 3.3 – Comparaison des performances de chiffrement/déchiffrement pour les quatre algorithmes étudiés.

Les opérations de chiffrement et de déchiffrement représentent des étapes essentielles dans tout système cryptographique, assurant respectivement la confidentialité et la récupération des données sécurisées.

La figure 3.3 présente une comparaison des performances des quatre algorithmes étudiés pour ces deux opérations. Les résultats obtenus révèlent une relation linéaire entre la taille des données et le temps d'exécution pour l'ensemble des configurations testées. L'analyse détaillée est la suivante :

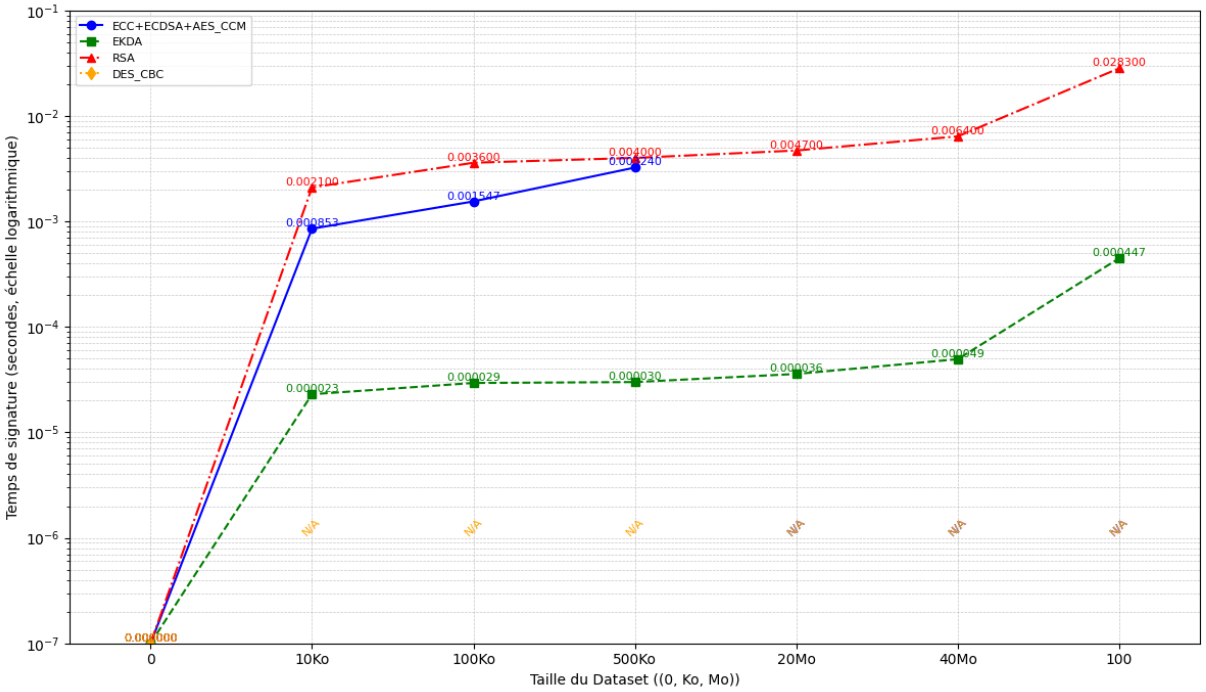
- **Solution hybride EKDA** : Démontre les meilleures performances globales avec une croissance linéaire stable du temps d'exécution.
- **ECC+ECDSA+AES_CCM** : Présente de bonnes performances jusqu'à 500 Ko, mais devient inutilisable au-delà en raison des limitations inhérentes au mode CCM.
- **DES_CBC** : Affiche des performances moyennes avec une progression régulière.
- **RSA** : Montre les temps d'exécution les plus élevés, particulièrement pour les grandes tailles de données.

Ces observations confirment l'efficacité de la solution hybride proposée qui maintient un équilibre optimal entre performance et sécurité.

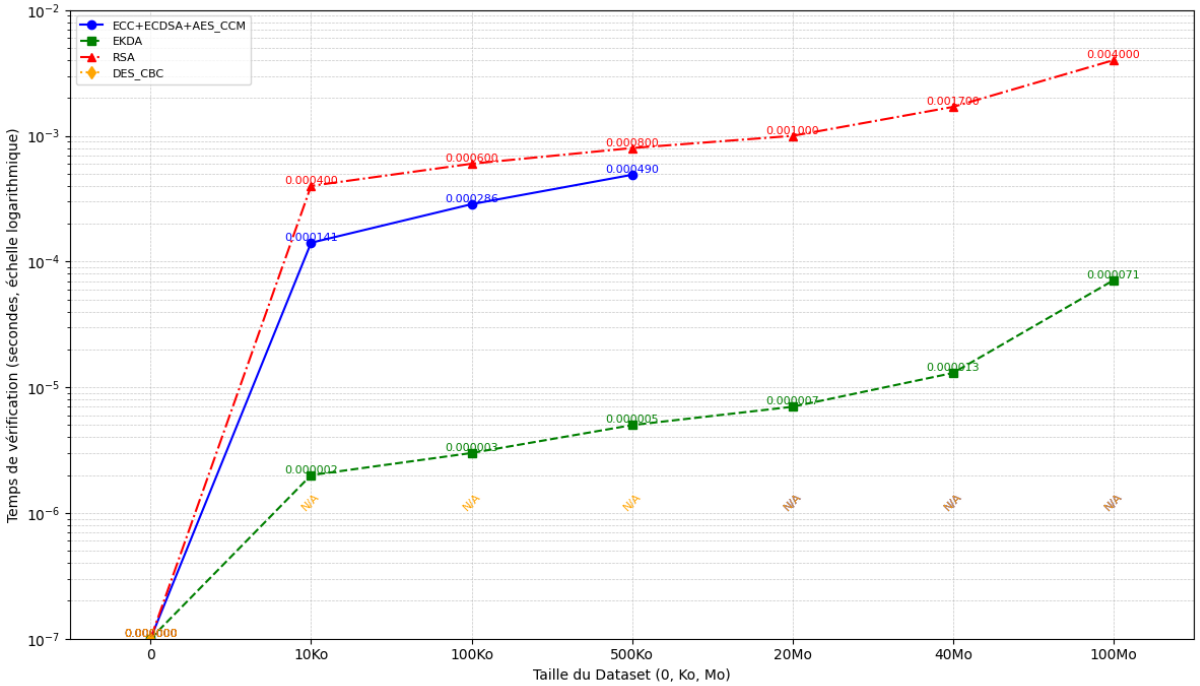
### 3.6.3 Génération et Vérification de Signatures

Le processus de génération et de vérification des signatures joue un rôle essentiel dans les systèmes cryptographiques, en garantissant l'authenticité et l'intégrité des données échangées.

La figure 3.4 présente une comparaison des performances des différents algorithmes de signature étudiés :




(a) Temps de génération des signatures en fonction de la taille des données.




(b) Temps de vérification des signatures en fonction de la taille des données.

FIGURE 3.4 – Temps de génération (a) et de vérification (b) des signatures selon la taille des données.

L'analyse comparative des performances révèle les observations suivantes :

A. **Génération de la signature**

- **Dilithium** : Présente les temps d'exécution les plus rapides et une excellente stabilité.
- **ECDSA** : Montre de bonnes performances mais limitées aux fichiers inférieurs ou égaux à 500 Ko a cause de mode de chiffrement CCM.
- **RSA** : Enregistre les temps d'exécution les plus élevés avec une croissance exponentielle.
- **DES** : Non applicable (algorithme de chiffrement symétrique uniquement).

B. **Vérification de la signature**

Les tendances restent similaires. Avec **Dilithium**, conserve son avantage en termes de rapidité et capacité de gérer des volumes croissants.

## 3.7 Performance du temps d'exécution moyen par algorithme

La figure 3.5 illustre une comparaison des temps d'exécution des algorithmes .

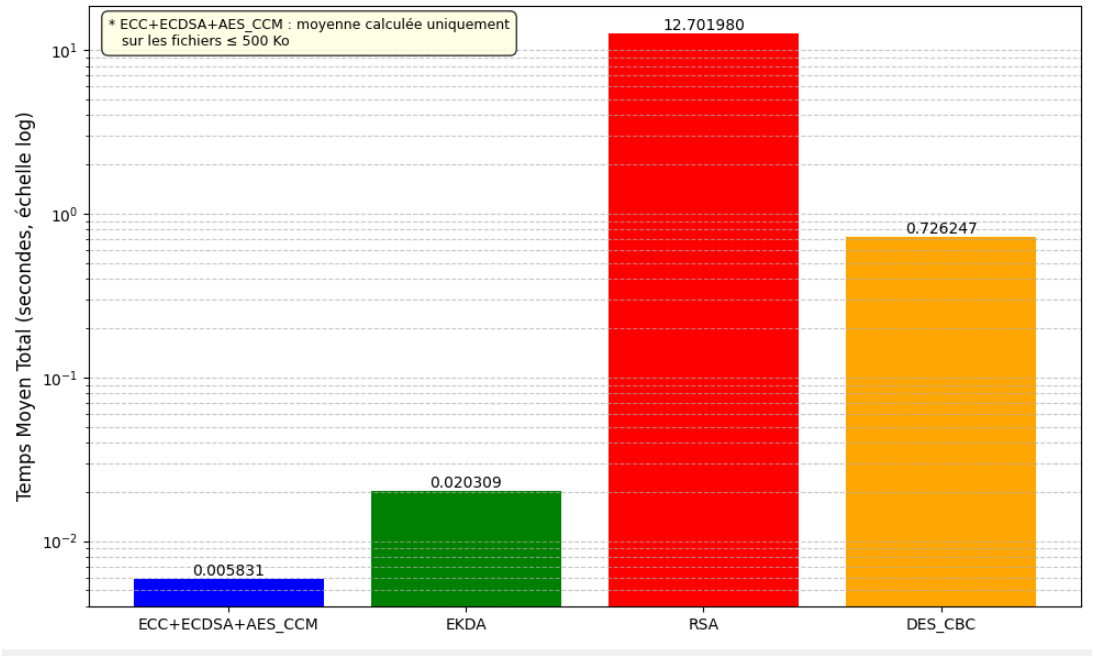



FIGURE 3.5 – Comparaison des Temps Moyens d'exécution des Algorithmes.

L'analyse du diagramme 3.5 met en évidence que l'algorithme proposé EKDA présente les meilleures performances en termes de temps moyen d'exécution, surpassant ainsi les autres algorithmes testés. L'algorithme DES_CBC se positionne en deuxième

place avec un temps d'exécution modéré, tandis que l'algorithme RSA enregistre le temps moyen le plus élevé. Cette lenteur est due à la complexité des calculs de factorisation des grands nombres dans l'algorithme RSA, ce qui le rend moins adapté aux applications demandant de la rapidité.

## 3.8 Utilisation du processeur CPU

Le graphique circulaire présenté à la Figure 3.6 illustre le partage proportionné de l'utilisation CPU moyenne des quatre algorithmes de chiffrement testés.

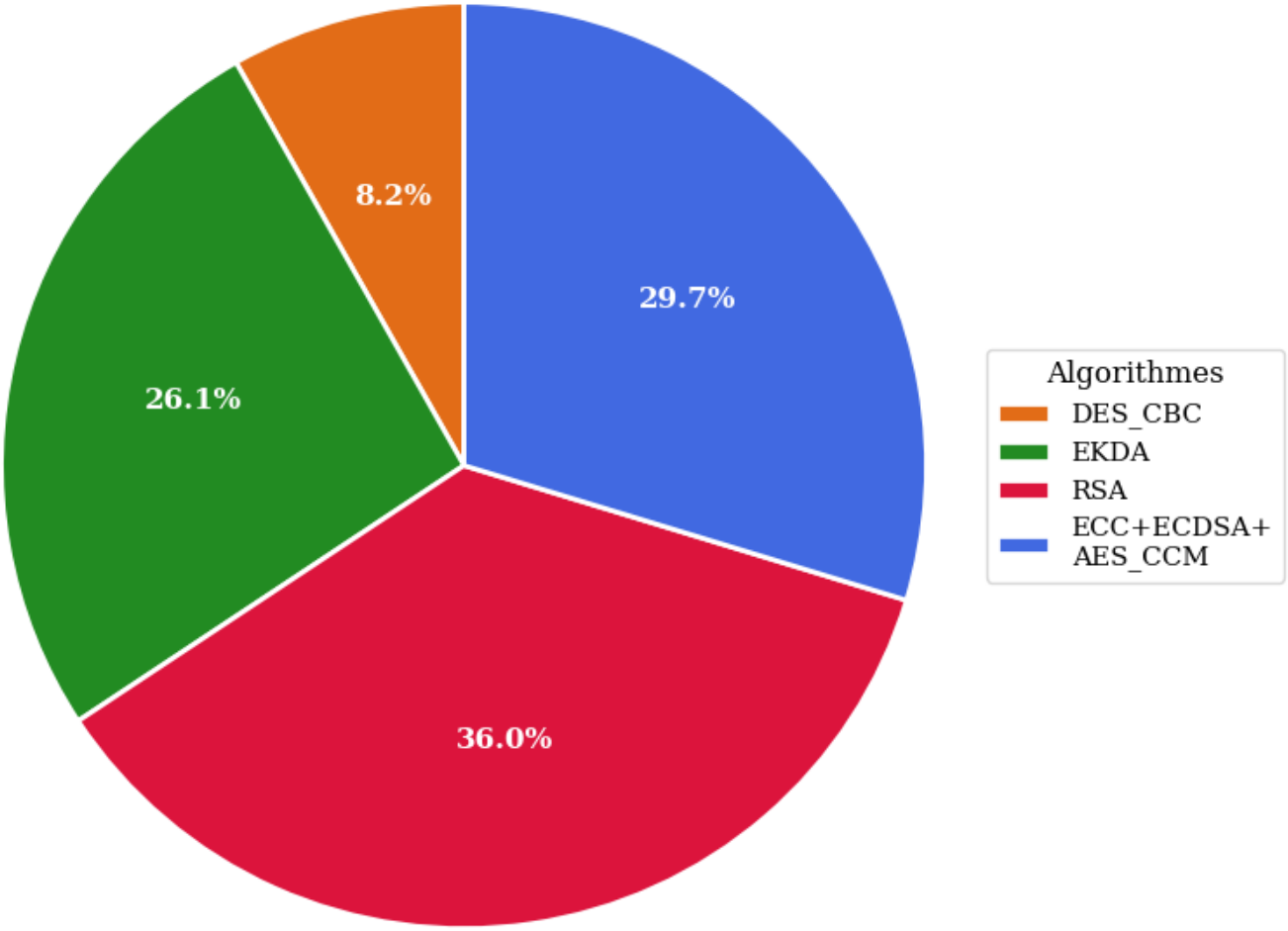



FIGURE 3.6 – Répartition de la consommation CPU par algorithme de chiffrement.

De l'analyse de la répartition CPU 3.6 nous remarquons quatre profils distincts :

- **DES_CBC (8.2%)** : Consommation minimale, idéal pour les environnements à ressources limitées.
- **EKDA (26.1%)** : Consommation modérée offrant le meilleur compromis performance/sécurité.

- **RSA (36%)** et **ECC+ECDSA+AES_CCM (29.7%)** : Consommation élevée et relativement proche.

## 3.9 Consommation mémoire RAM

Le graphique linéaire 3.7 présente la quantité de mémoire vivre (RAM) utilisé par un algorithme pendant son exécution en fonction de taille des fichiers traités (de 10Ko à 100Ko).

**Analyse Graphique et Observations** Le graphique linéaire 3.7 présente l'évolution de la consommation mémoire des quatre algorithmes de chiffrement en fonction de la taille des fichiers traités, allant de 10 Ko à 100 Mo.

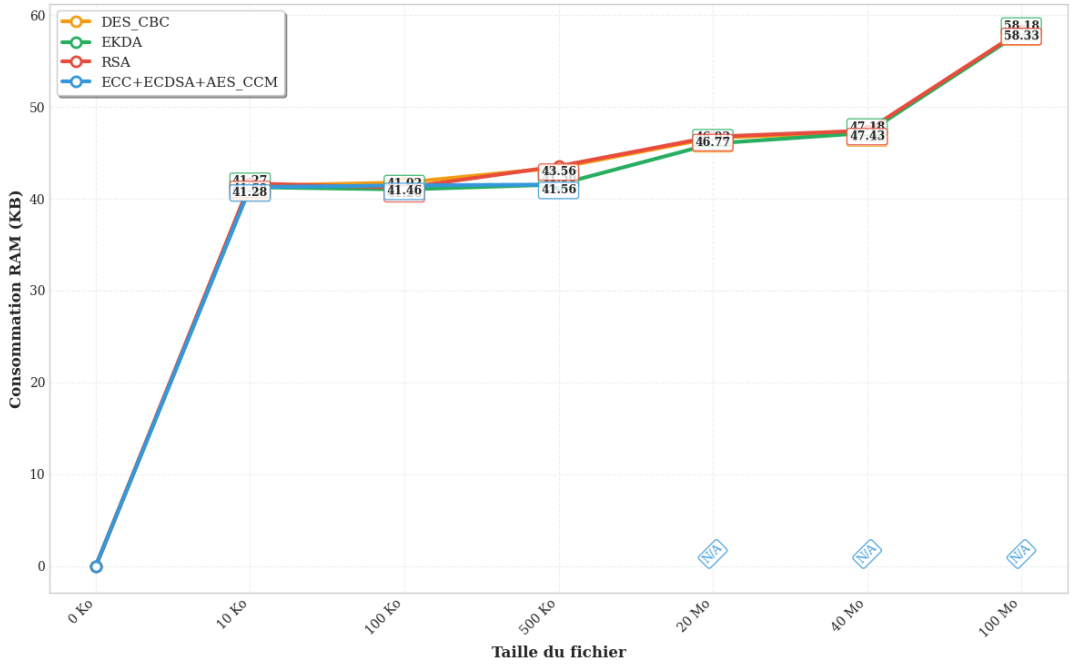



FIGURE 3.7 – Consommation de RAM selon la taille du fichier.

L'analyse souligne une tendance croissante de la consommation RAM avec l'augmentation de la taille des fichiers pour tous les algorithmes testés. Cette relation positive s'explique par la nécessité d'allouer davantage de mémoire tampon pour traiter des volumes de données plus importants.

Le graphique en barres 3.8 ci-dessous présente le classement des algorithmes selon leur consommation RAM moyenne, permettant une comparaison directe de leur efficacité globale.

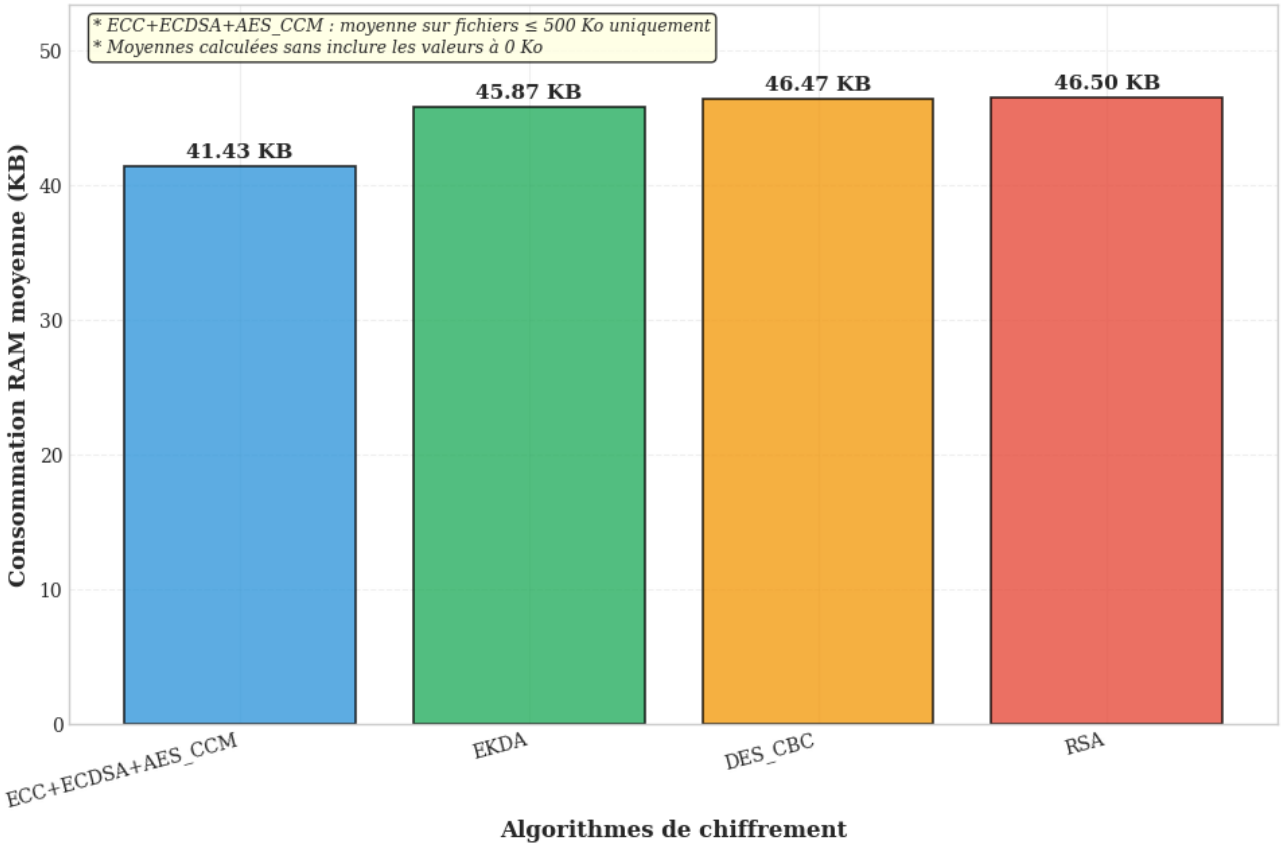



FIGURE 3.8 – Classement par consommation RAM moyenne.

L'analyse de la figure 3.8 montre que les algorithmes **DES_CBC** et **RSA** ont une bonne scalabilité avec une utilisation mémoire stable d'environ 46 à 48 KB pour des fichiers allant de 1 à 100 Mo. De son côté, la combinaison **ECC+ECDSA+AES_CCM** est efficace pour les petits fichiers (environ 41 KB pour des tailles inférieures ou égales à 500 Ko), mais elle n'est pas adaptée aux grands volumes à cause de sa structure cryptographique complexe. En revanche, la solution hybride **(EKDA)** offre de bonnes performances mémoire (entre 41 et 46 KB), ce qui en fait un choix optimal pour les applications modernes. Ces résultats montrent qu'il est nécessaire d'abandonner progressivement les anciens protocoles (comme DES et RSA seuls) au profit de solutions hybrides plus robustes, tout en gardant ECC+ECDSA+AES_CCM pour les systèmes légers embarqués.

## 3.10 Mesures de Sécurité

Cette section présente une étude détaillée des faiblesses potentielles tant face aux attaques classiques qu'aux attaques quantiques.

### 3.10.1 Analyse des Vulnérabilités face aux Attaques Classiques

Les mécanismes cryptographiques testés révèlent certaines faiblesses face aux attaques classiques ci-dessous. :

- **Scénario A (EKDA)**
  Résistant aux attaques classiques grâce à l'intégration de primitives modernes (Kyber, Dilithium) conçues pour contrer les faiblesses connues. Toutefois, une mauvaise gestion des nombres aléatoires ou des canaux auxiliaires pourrait compromettre sa sécurité si les contre-mesures ne sont pas appliquées correctement.
- **Scénario B (DES-CBC)**
  Extrêmement vulnérable aux attaques par force brute en raison de la faible taille de clé (56 bits), ce qui rend l'algorithme obsolète face à la puissance des processeurs modernes. De plus, il est sensible aux attaques par différences (cryptanalyse différentielle) et linéaires, connues depuis des décennies.
- **Scénario C (ECC+ECDSA+AES_CCM)**
  Vulnérable aux attaques par canaux auxiliaires (temps d'exécution, cache timing) si l'implémentation n'est pas protégée. De plus, le mode CCM présente des limitations pour les grandes tailles de données et pourrait être exploité en cas de nonce réutilisé ou mal généré. La sécurité dépend également fortement de la robustesse de la fonction de hachage utilisée (SHA-2).
- **Scénario D (RSA)**
  Très exposé aux attaques par force brute en cas d'utilisation de clés insuffisamment longues **(inférieur à 2048 bits)**. Également vulnérable aux attaques par canaux auxiliaires (par exemple via des attaques de type « timing » ou « power analysis ») si aucune contre-mesure n'est appliquée. Le temps d'exécution élevé peut aussi être exploité dans des attaques par déni de service (DoS).

### 3.10.2 Analyse de la Vulnérabilité aux Attaques Quantiques

Les configurations cryptographiques classiques évaluées présentent des vulnérabilités critiques :

- **Scénario C (ECC+ECDSA+AES_CCM)**
  Vulnérable car l'algorithme de Shor peut résoudre efficacement le problème du logarithme discret sur courbes elliptiques, compromettant totalement l'échange de clés ECDH ainsi que les signatures ECDSA.

- **Scénario D (RSA)**
  Subit une compromission complète puisque l'algorithme de Shor permet de factoriser les grands nombres premiers utilisés par RSA.
- **Scénario B (DES-CBC)**
  Bien que moins directement affectée par l'algorithme de Shor, cette configuration reste vulnérable aux attaques par l'algorithme de Grover, réduisant ainsi sa sécurité effective de 56 à 28 bits.
- **Scénario A (EKDA)** Cette configuration assure une sécurité renforcée grâce à son approche hybride combinant primitives classiques et post-quantiques. Ce modèle garantit la confidentialité et l'intégrité des données tant qu'au moins l'une des composantes reste robuste face aux attaques. La présence de mécanismes post-quantiques (Kyber pour l'échange de clés et Dilithium pour la signature) assure une protection durable contre les menaces futures, notamment celles provenant de l'informatique quantique, conformément aux standards actuels du **NIST**.

## 3.11 Synthèse des Résultats obtenus

D'après l'analyse comparative approfondie présentée dans ce chapitre, la proposition hybride post-quantique **EKDA** se distingue comme la configuration optimale, offrant le meilleur équilibre entre performance et sécurité. Voici les principaux points qui justifient cette conclusion :

1. **Haute performance temporelle**
   - La solution hybride affiche le **temps d'exécution moyen le plus faible** (0.020 s) parmi tous les algorithmes testés, surpassant largement DES_CBC (0.73 s) et RSA (12.70 s). Cette rapidité est cruciale pour les applications nécessitant des traitements en temps réel.
   - Elle démontre une **scalabilité remarquable**, avec des performances stables quelle que soit la taille des données (de 10 Ko à 100 Mo), contrairement à ECC+ECDSA+AES_CCM qui devient inefficace pour des fichiers dépassant 500 Ko.
2. **Efficacité des Algorithmes Clés**
   - **AES en mode GCM** s'est révélé être le choix optimal pour le chiffrement symétrique, combinant rapidité et sécurité, notamment grâce à sa capacité à gérer des volumes de données importants sans compromettre les performances.

- **Kyber (pour l'échange de clés)** et **Dilithium (pour les signatures)** offrent des temps de génération et de vérification compétitifs, tout en étant résistants aux attaques quantiques.

3. **Consommation Modérée des Ressources**
   - **CPU** : Bien que la solution hybride présente une consommation CPU plus élevée que DES_CBC (26.1 % contre 8.2 %), elle reste bien inférieure à RSA (36 %) et ECC+ECDSA+AES_CCM (29.7 %). Cette consommation est justifiée par sa robustesse et sa polyvalence.
   - **RAM** : La solution hybride maintient une **utilisation mémoire stable et raisonnable** (entre 41 et 48 KB), comparable aux autres algorithmes, tout en offrant une sécurité supérieure.
4. **Résistance aux Attaques Classiques et Quantiques**
   - La solution hybride intègre des primitives modernes **résistantes aux attaques quantiques** (Kyber et Dilithium), contrairement à RSA et ECDSA qui sont vulnérables à l'algorithme de Shor.
   - Elle est également **robuste face aux attaques classiques** (force brute, canaux auxiliaires), grâce à des mécanismes de protection avancés et une gestion rigoureuse des clés.
5. **Limitations des autres approches**
   - **DES-CBC** : Obsolète en raison de sa faible taille de clé (56 bits) et de sa vulnérabilité aux attaques par force brute.
   - **RSA** : Trop lent et vulnérable aux attaques quantiques (Shor), ce qui le rend inadapté aux besoins modernes.
   - **ECC+ECDSA+AES_CCM** : Performant pour les petits fichiers, mais inefficace pour les grandes tailles de données et vulnérable aux attaques quantiques.

## 3.12 Conclusion

La solution hybride proposée s'impose comme la meilleure alternative, en alliant rapidité, sécurité et adaptabilité. Elle répond efficacement aux exigences des systèmes actuels tout en anticipant les menaces des ordinateurs quantiques de demain. Les résultats obtenus démontrent qu'elle offre d'excellentes performances, une consommation modérée des ressources et une forte résistance face aux attaques classiques et quantiques. Contrairement aux approches traditionnelles telles que ECC-AES, DES ou RSA, cette solution se distingue par son évolutivité et sa robustesse accrues, ce qui en fait un choix idéal pour garantir des communications sécurisées à l'ère post-quantique.

# Conclusion Générale

La montée en puissance de l'informatique quantique remet en question la robustesse des mécanismes de sécurité classiques, notamment ceux fondés sur RSA et ECC, et rend insuffisante une protection reposant sur une seule famille d'algorithmes. Pour répondre à ce défi, ce mémoire a proposé EKDA, une architecture cryptographique hybride innovante alliant la légèreté et l'efficacité de la cryptographie à courbes elliptiques (ECDH) aux garanties post-quantiques de Kyber (KEM) et Dilithium (signature), tout en recourant à AES-GCM pour le chiffrement symétrique authentifié.

Nous avons implémenté ce modèle dans un prototype capable de traiter divers types de fichiers, puis mené une évaluation expérimentale exhaustive. Les résultats démontrent que EKDA atteint un compromis satisfaisant entre sécurité renforcée, résistante à la fois aux attaques classiques et quantiques et performance pratique, en termes de temps de traitement, d'empreinte mémoire et de taille de clé.

Toutefois, certaines limites subsistent : la configuration initiale du système demeure complexe et la gestion fine des paramètres cryptographiques peut se révéler coûteuse dans des environnements très contraints. De plus, bien que les tests en laboratoire soient encourageants, une validation en conditions réelles est indispensable pour confirmer la maturité de la solution.

Plusieurs perspectives s'ouvrent pour prolonger ce mémoire :

- Optimiser EKDA pour le chiffrement de flux en temps réel et le traitement de très gros volumes de données.
- Étendre la chaîne modulaire à d'autres primitives post-quantique en conservant les interfaces existantes.
- Renforcer la confiance dans la sécurité par une formalisation mathématique et des preuves de réduction adaptées.
- Déployer EKDA dans des cas d'usage concrets (messagerie d'entreprise, IoT, cloud sécurisé) afin d'éprouver son intégration, son ergonomie et sa robustesse à grande

échelle.

En somme, EKDA représente une réponse prometteuse face aux enjeux de la cryptographie dans l'ère post-quantique, posant les bases d'une architecture évolutive, robuste et adaptée aux exigences futures en matière de cybersécurité.

# BIBLIOGRAPHIE


[1] C. THUILLET, "Implantations cryptographiques sécurisées et outils d'aide à la validation des contremesures contre les attaques par canaux cachés," Thèse de doctorat, Université BORDEAUX I, France, mars 2012.

[2] R. DUMONT, *Cryptographie et Sécurité informatique*, Faculté des Sciences Appliquées, 2010.

[3] B. BRAHIM, "Cryptographie : Approche Quantique," Promotion 2016/2017, Option Télécommunication et réseaux, Mémoire de Master, Université Mouloud Mammeri de Tizi-Ouzou, Tizi-Ouzou, Algérie, 2017.

[4] U. BHARGAVA, A. SHARMA, R. CHAWLA et P. THAKRAL, "A new algorithm combining substitution transposition cipher techniques for secure communication," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, IEEE, 2017, p. 619-624.

[5] U. BHARGAVA, P. SHARMA, S. SHARMA et S. SHARMA, "A new algorithm combining substitution, transposition cipher techniques for secure communication," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, IEEE, 2017, p. 1023-1027.

[6] P. GUILLOT, *Courbes elliptiques : Une présentation élémentaire pour la cryptographie*. 11 rue Lavoisier, 75008 Paris : Ellipses, 2010, ISBN : 2-7462-1260-9.

[7] M. NIANE, "Conventional Hill Algorithm : From Classical Cryptography to Modern Cryptography," 2022.

[8] T. EBRAHIMI, F. LÊPRÉVOST et B. WARUSFEL, *Cryptographie et sécurité des systèmes et réseaux*. Lausanne : PPUR Presses Polytechniques Universitaires Romandes, 2006, ISBN : 978-2-88074-657-1.

[9] G. HERBIET, *Cryptographie classique*, Cours de cryptographie, Université de Liège, 2022.

[10] D. KAHN, *The Codebreakers : The Story of Secret Writing*. London : Weidenfeld & Nicolson, 1968.

[11] D. KAHN, *The Story of Secret Writing*. London : Weidenfeld & Nicolson, 1968.

[12] L. MOUSSAOUI et S. CHOUAICHIA, "Étude et simulation d'un cryptosystème basé sur l'algorithme AES-GCM : Application au cryptage des images médicales," Mémoire de master, Université 8 Mai 1945 – Guelma, Algérie, juin 2023.

[13] C. THUILLET, "Implantations cryptographiques sécurisées et outils d'aide à la validation des contremesures contre les attaques par canaux cachés," Thèse de doctorat, Université Bordeaux I, France, mars 2012.

[14] R. BENIANI, "Sécurité des Images Numériques Compressées JPEG," Thèse de doctorat, Université Djillali Liabès de Sidi Bel Abbès, Algérie, 2019.

[15] COLLECTIF, *Paiements électroniques sécurisés*. France : Dunod, 2009.

[16] A. A. ISMAILI et A. MOUSSA, "Self-partial and dynamic reconfiguration implementation for AES using FPGA," *International Journal of Computer Science Issues (IJCSI)*, t. 1, p. 33-40, août 2009.

[17] A. GONTIER, "Utilisation de solveurs génériques pour la cryptanalyse de chiffrements symétriques," Thèse présentée et soutenue à Rennes en novembre 2023, Thèse de doctorat, Université de Rennes, Rennes, France, 2023.

[18] M. PETIT, *Énergie, Entropie, Information, Cryptographie et Cybersécurité*. France : Éditions Quae, 2022.

[19] S. CHOUAICHIA, "Etude et simulation d'un crypto-système basé sur l'algorithme AES-GCM," Master's thesis, Université de Guelma, 2023.

[20] ANSSI, *Avis scientifique et technique sur la migration vers la cryptographie post-quantique*, Agence nationale de la sécurité des systèmes d'information (ANSSI), 2022.

[21] J.-J. KAYEMBE, *La Sécurité Informatique au Congo*. France : L'Harmattan, 2010.

[22] L. SALVAIL, *Cryptographie avancée, Notes de cours*, 2014.

[23] D. N.YESSAD, *Chapitre 1. Introduction à la sécurité des systèmes d'information*, 2023.

[24] V. UNLIMITED. "Replay Attack : Understanding and Prevention." Consulté le 16/04/2024. (2023), adresse : https://www.vpnunlimited.com/fr/help/cybersecurity/replay-attack.

[25] CLOUDFLARE. "Attaques de phishing : protection des données personnelles." Consulté le 16/04/2024. (2023), adresse : https://www.cloudflare.com/fr-fr/learning/access-management/phishing-attack/.

[26] FORTINET. "Ransomware : Définition et Prévention." Consulté le 18/06/204. (2023), adresse : https://www.fortinet.com/fr/resources/cyberglossary/ransomware.

[27] CLOUDFLARE. "Injection SQL : Prévention et Protection." Consulté le 21/06/2024. (2024), adresse : https://www.cloudflare.com/fr-fr/learning/security/threats/how-to-prevent-sql-injection/.

[28] I. SOCIETY. "Qu'est-ce qu'une attaque de l'homme du milieu (MITM) ?" le 21/06/2024. (2019), adresse : https://www.internetsociety.org/fr/blog/2019/11/quest-ce-quune-attaque-de-lhomme-du-milieu-mitm/.

[29] WAYTOLEARNX. "Différence entre attaque active et attaque passive." Consulté le 21/06/2024. (2018), adresse : https://waytolearnx.com/2018/07/difference-entre-attaque-active-et-attaque-passive.html.

[30] COMMISSION NATIONALE DE L'INFORMATIQUE ET DES LIBERTÉS (CNIL). "Attaque par force brute." Consulté le 21/06/2024. (2023), adresse : https://www.cnil.fr/fr/definition/force-brute-attaque-informatique.

[31] JEDHA. "Attaque par dictionnaire : Définition, Étapes et Protection." Consulté le 21/06/2024. (2023), adresse : https://www.jedha.co/formation-cybersecurite/attaque-par-dictionnaire-definition-etapes-et-protection.

[32] V. JUSTICE. "Que dit la loi sur l'utilisation des caméras espions ?" Consulté le 25/06/2024. (2023), adresse : https://www.village-justice.com/articles/que-dit-loi-sur-utilisation-des-cameras-espion,43009.html.

[33] R. de l'Université du QUÉBEC, *Définition d'un mot de passe robuste (Version 1.2)*, version 1.2, Consulter le 28/04/2024, Réseau de l'Université du Québec.

[34] F. Z. BOUDERBALA, "Approche de sécurisation des systèmes informatiques basée sur les comportements," Thèse de doctorat en Informatique, Thèse de doctorat, Université de Blida, Blida, Algérie, 2018.

[35] GEEKSFORGEEKS, *PGP Authentication and Confidentiality*, Consulté le 25/05/2024, 2023. adresse : https://www.geeksforgeeks.org/pgp-authentication-and-confidentiality.

[36] TECHNO-SCIENCE.NET. "Secure Shell - Définition." Consulté le 02/04/2025, Techno-Science.net. (2023), adresse : https://www.techno-science.net.

[37] Y. SHOU, "Cryptographie sur les courbes elliptiques et tolérance aux pannes dans les réseaux de capteurs," Thèse de doctorat, thèse de doct., Université de Franche-Comté, Besançon, France, 2014.

[38] E.-I. BARTZIA, "A Formalization of Elliptic Curves for Cryptography," anglais, Thèse de doctorat, thèse de doct., Université Paris-Saclay, 2017.

[39] ED-DIAMOND, "Standardisation des courbes elliptiques : à qui faire confiance ?" *MISC*, Consulté le 13/05/2025. adresse : https://connect.ed-diamond.com/MISC/mischs-013/standardisation-des-courbes-elliptiques-a-qui-faire-confiance.

[40] A. LOISEAU, "Implémentation légère et sécurisée pour la cryptographie sur courbes elliptiques pour l'Internet des Objets," Consulté le 13/05/2025, Thèse de doctorat, Université de Lyon, France, 2019.

[41] V. S. MILLER, "Use of Elliptic Curves in Cryptography," Lecture Notes in Computer Science, t. 218, p. 417-426, 1985. DOI : 10.1007/3-540-39799-X_31.

[42] N. KOBLITZ, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, t. 48, n° 177, p. 203-209, 1987. DOI : 10.2307/2007884.

[43] D. JOHNSON et A. MENEZES, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *International Journal of Information Security*, t. 1, n° 1, p. 36-63, 2001, Analyse complète de l'ECDSA et de sa sécurité. DOI : 10.1007/s102070100002.

[44] A. A. ALHAJ, A. ALRABEA et O. A. A. JAWABREH, "Efficient and secure data transmission : cryptography techniques using ECC," *Indonesian Journal of Electrical Engineering and Computer Science*, t. 36, n° 1, p. 486-492, oct. 2024, ISSN : 2502-4752.

[45] P. GAYATHRI, S. UMAR, G. SRIDEVI, N. BASHWANTH et R. SRIKANTH, "Hybrid Cryptography for Random-key Generation based on ECC Algorithm," *International Journal of Electrical and Computer Engineering (IJECE)*, t. 7, n° 3, p. 1293-1298, 2017.

[46] A. MITRA, S. CHAKRABARTY et P. MITRA, "Elliptic Curve Cryptosystem for Email Encryption," *International Journal of Computer and Communication Technology*, t. 1, n° 4, p. 229-233, 2010.

[47] C. NEELIMA et C. SUNEETHA, "Minimising the Cipher Length of Elliptic Curve Cryptography Basing on Chebyshev Polynomial," *Journal of Information Systems Engineering and Management*, t. 10, n° 38s, p. 294-303, 2025, ISSN : 2468-4376.

[48] B. MURTHY et I. S. V. SUBRAMANYAM, "SECURE EMAIL SERVICES USING ECC," *International Journal of Engineering Science and Advanced Technology (IJESAT)*, t. 24, n° 07, p. 15-24, juill. 2024, ISSN : 2250-3676.

[49] C.-H. YANG, T.-Y. KUO, T. AHN et C.-P. LEE, "Design and Implementation of a Secure Instant Messaging Service based on Elliptic-Curve Cryptography," *Journal of Computers*, t. 18, n° 4, p. 31-38, 2008.

[50] D. N. PURNAMASARI, A. SUDARSONO et P. KRISTALINA, "Secure e-Health Record System Using Identity-based Encryption with Embedded Key," *International Journal on Advanced Science, Engineering and Information Technology*, t. 9, n° 5, p. 1495-1504, 2019, ISSN : 2088-5334.

[51] X. LIN, "The Application of Elliptic Curve Cryptography in Secure E-mail System," *International Conference on Advanced Material Science and Environmental Engineering (AMSEE)*, t. 1, 2016.

[52] D. JAGADESSH, K. DINESH, T. S. REDDY, B. P. MAHAJAN, U. AMRUTHA et S. S. V., "Optimized Cryptographic Techniques for Image Security Using ECC and SHA," *International Journal of Science & Engineering*, t. 8, n° 1, p. 53-58, 2025, ISSN : 2456-3293.

[53] R. AVANZI, J. BOS, L. DUCAS et al., "Algorithm Specifications And Supporting Documentation (version 3.02)," NIST Post-Quantum Cryptography Project, rapp. tech., août 2021, Kyber Round 3 Specification.

[54] Y. KIM, J. SONG, T.-Y. YOUN et S. C. SEO, "Crystals-Dilithium on ARMv8," *Security and Communication Networks*, t. 2022, Article ID 5226390, 12 pages, 2022.

[55] H. KRAWCZYK et P. ERONEN, *RFC 5869 : HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*, 2010.

[56] M. PARMAR, *Structure your code better in Google Colab with text and code cells*, https://miteshparmar1.medium.com/structure-your-code-better-in-google-colab-with-text-and-code-cells-b6fa73feec20, Consulté le 24 mai 2025, 2020.

## RÉSUMÉ

Ce mémoire s'inscrit dans le domaine de la sécurité de l'information à l'ère numérique, où les systèmes cryptographiques classiques sont confrontés à des menaces croissantes, notamment celles liées à l'émergence de l'informatique quantique. Dans ce contexte, nous proposons une solution cryptographique hybride intégrant des algorithmes classiques (ECC, AES-GCM) et post-quantiques (Kyber, Dilithium), afin de garantir la confidentialité, l'authenticité et la robustesse des échanges de données. Notre objectif est de concevoir un système capable de répondre aux exigences actuelles et futures en matière de sécurité, tout en assurant des performances acceptables en termes de temps d'exécution et de consommation de ressources. La méthodologie adoptée repose sur une étude théorique approfondie, l'implémentation d'un prototype fonctionnel et une évaluation expérimentale des performances du système proposé.



**Mots-clés :** Cryptographie hybride, évaluation des performances, signature numérique, cryptographie post-quantique, ECC, Kyber, Dilithium, AES-GCM.



## ABSTRACT

This thesis is situated within the field of information security in the digital era, where traditional cryptographic systems face growing threats, particularly those posed by the rise of quantum computing. In this context, we propose a hybrid cryptographic solution combining classical algorithms (ECC, AES-GCM) with post-quantum algorithms (Kyber, Dilithium) to ensure data exchange confidentiality, authenticity, and robustness. Our objective is to design a system capable of meeting current and future security demands while maintaining acceptable performance in terms of execution time and resource consumption. The adopted methodology is based on a thorough theoretical study, the implementation of a functional prototype, and an experimental evaluation of the proposed system's performance.



**Keywords:** Hybrid cryptography, performance evaluation, digital signature, post-quantum cryptography, ECC, Kyber, Dilithium, AES-GCM.