CHARM Cryptosystème Hybride Avancé pour les Réseaux Mobiles

4.1 Introduction

L'objectif de notre thème consiste à présenter une nouvelle méthode de Cryptographie pour les Réseaux Mobile (par la proposition d'un Cryptosystème Hybride). Le Cryptosystème proposé et développé est dédié aux unités de calculs mobiles qui ont des caractéristiques particulières, une puissance de calcul limitées, une faible capacité de stockage et une source d'énergie autonome.

Après avoir présenté une étude détaillée sur la méthode de cryptographie symétrique qui est AES (Advanced Encryption Standard) et la cryptographie à base des courbes elliptiques qui servent comme support théorique pour ce chapitre. Nous allons présenter notre méthode de chiffrement hybride qui associe entre les deux méthodes étudiées et qui combine les propriétés et les avantages de chacune. On l'applique à la transmission des données. Afin d'améliorer les performances en terme de temps de calcul et le niveau de sécurité.

4.2 Présentation du cryptosystème

4.2.1 Cryptosystème Hybride

Un cryptosystème hybride consiste à utiliser les avantage des chiffrements symétrique et asymétrique tels que :

- La rapidité d'un système symétrique.
- La possibilité de transmettre la clé secrète par un cryptosystème asymétrique.

La plus part des cryptosystèmes hybrides procèdent de la manière suivante :

Une clé privée est générée pour l'algorithme symétrique. L'algorithme de chiffrement symétrique est ensuite utilisé pour chiffrer le message. La clé privée est échangée entre l'émetteur et le récepteur grâce à l'algorithme de chiffrement asymétrique. Il suffit ensuite d'échanger la clé privée et envoyer le message chiffré avec l'algorithme symétrique. Le destinataire utilise la clé secrète échangée pour le déchiffrement symétrique et retrouve le message.

4.2.2 Choix des algorithmes

- ✓ **Chiffrement symétrique :**

Pour la cryptographie symétrique, nous avons choisi l'algorithme AES (Advanced Encryption Standard), que nous avons vu au chapitre précédent est un chiffrement par blocs. Il traite des messages de 128 bits avec une clé de 128,192 ou 256 bits, et fournit un cryptogramme de 128 bits en sortie.

L'emploi de la cryptographie chaotique par l'utilisation de la fonction logistique comme un générateur pseudo-aléatoire de SBOX et INVSBOX de l'AES, a pour équation suivante:

$$Xn{+}1{=}\ \mu Xn(1{-}Xn) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4.1)$$

Les paramètres initiales ($X0$, μ) sont décrit dans la section suivante.

- ✓ **Chiffrement asymétrique :**

Afin de permettre un échange sécurisé de clé, un mécanisme de chiffrement à clé publique « Le **E-C-C**: **E**lliptic **C**urve **C**ryptography» basé sur le logarithme discret est privilégié dans notre cryptosystème.

4.3 Conception de cryptosystème

Le cryptosystème proposé est constitué de trois phases suivantes :

- L'échange de clé principale.
- Génération pseudo-aléatoire de SBOX et INVSBOX.
- Chiffrement/Déchiffrement AES.

4.3.1 L'échange de clé principale

Pour générer et échanger une clé en commun entre l'émetteur et le récepteur, on utilise le protocole d'échange de clé de Diffie-Hellmann basé sur les courbes elliptiques de la manière suivante :

- l'émetteur et le récepteur choisissent une courbe elliptique E définie sur un corps fini $\mathbf{F_q}$ tel que le logarithme discret soit difficile à résoudre.
- Ils choisissent aussi un point P ϵ E($\mathbf{F_q}$) tel que le sous-groupe généré par P a un ordre de grande taille. (En général, la courbe E et le point P sont choisis de manière a ce que l'ordre soit un grand nombre premier).
- l'émetteur choisit un nombre entier secret a, calcule P_a = aP et envoie P_a au récepteur.

- Le récepteur choisit un nombre entier secret b, calcule P_b = bP et envoie Pb à l'émetteur.
- l'émetteur calcule aP_b= abP.
- Le récepteur calcule bP_a = baP.
- l'émetteur et le récepteur utilisent une méthode quelconque connue pour extraire une clé secrète de « abP ». Par exemple, ils peuvent utiliser les premiers 128 bits de la première coordonnée de « abP » comme clé de chiffrement symétrique AES, et les derniers 14 bits de la première coordonnée de « abP » comme vecteur initial $X0$ de la fonction logistique et les premiers 14 bits de la deuxième coordonnée de « abP » comme clé de la fonction logistique.

La figure 3.14 illustre l'application de la méthode de Diffie-Hellman aux courbes elliptiques pour échanger la clé.

4.3.2 Génération pseudo-aléatoire de SBOX et INVSBOX avec la fonction logistique

4.3.2.1 La fonction logistique

La fonction logistique est l'une des fonctions chaotiques les plus connues, définies comme suit :

$$Xn+1= \mu Xn\ (1-Xn)$$

Où Xn_{+1} et Xn sont respectivement les sorties actuelles et précédentes de la fonction, les deux tombent dans la plage de [0,1] tandis que μ est le paramètre de contrôle dans la plage de [0,4]. La plage du comportement chaotique de la fonction logistique est limitée à [3.57,4] ce qui conduit à un espace clé limité.

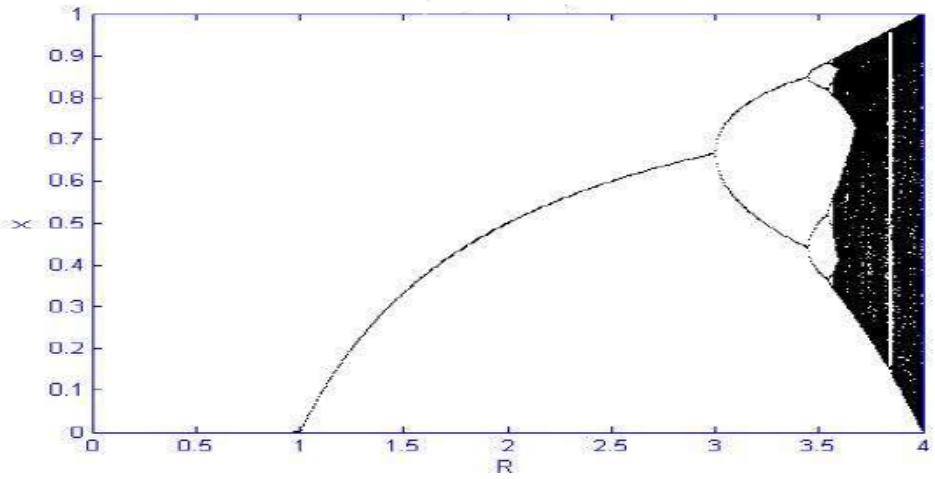

Figure 4.1 : comportement de la fonction logistique.

4.3.2.2 Génération pseudo-aléatoire de SBOX

Après avoir extraire les trois clé K1, K2, K3, On utilise la fonction logistique comme un générateur pseudo-aléatoire de SBOX et INVSBOX de l'AES, a pour équation :

$$Xn{+}1{=}\ \mu Xn\ (1{-}Xn)$$

Les paramètres initiales sont : $X0$ = K2 , μ = K3.

Le générateur pseudo-aléatoire basé sur la fonction logistique est représenté comme suit :

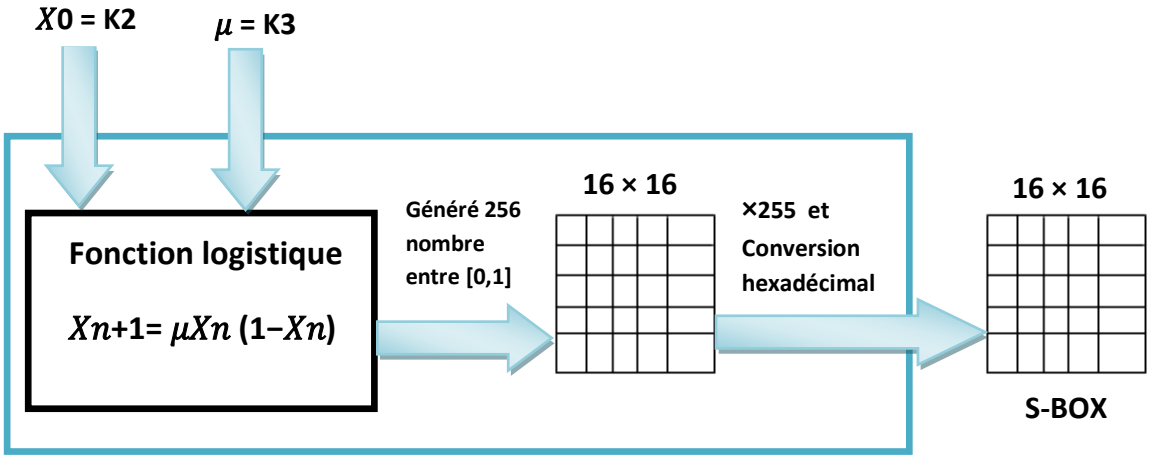

Figure 4.2 : Générateur pseudo-aléatoire de S-box basé sur la fonction logistique.

4.3.2.3 Génération pseudo-aléatoire de INVSBOX

Pour le processus de déchiffrement, on a besoin d'INVSBOX qui est extrait de SBOX avec une fonction inverse :

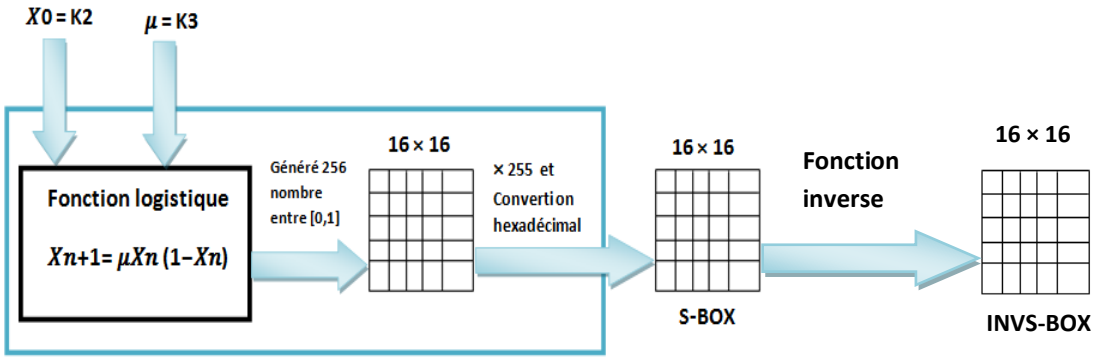

Figure 4.3 : Extraction d'INVS-BOX à partir de S-BOX.

4.3.3 Chiffrement/Déchiffrement AES

4.3.3.1 Chiffrement

Afin de générer SBOX avec la fonction logistique, la clé K1 est utilisée dans le chiffrement symétrique AES, comme l'indique la figure suivante :

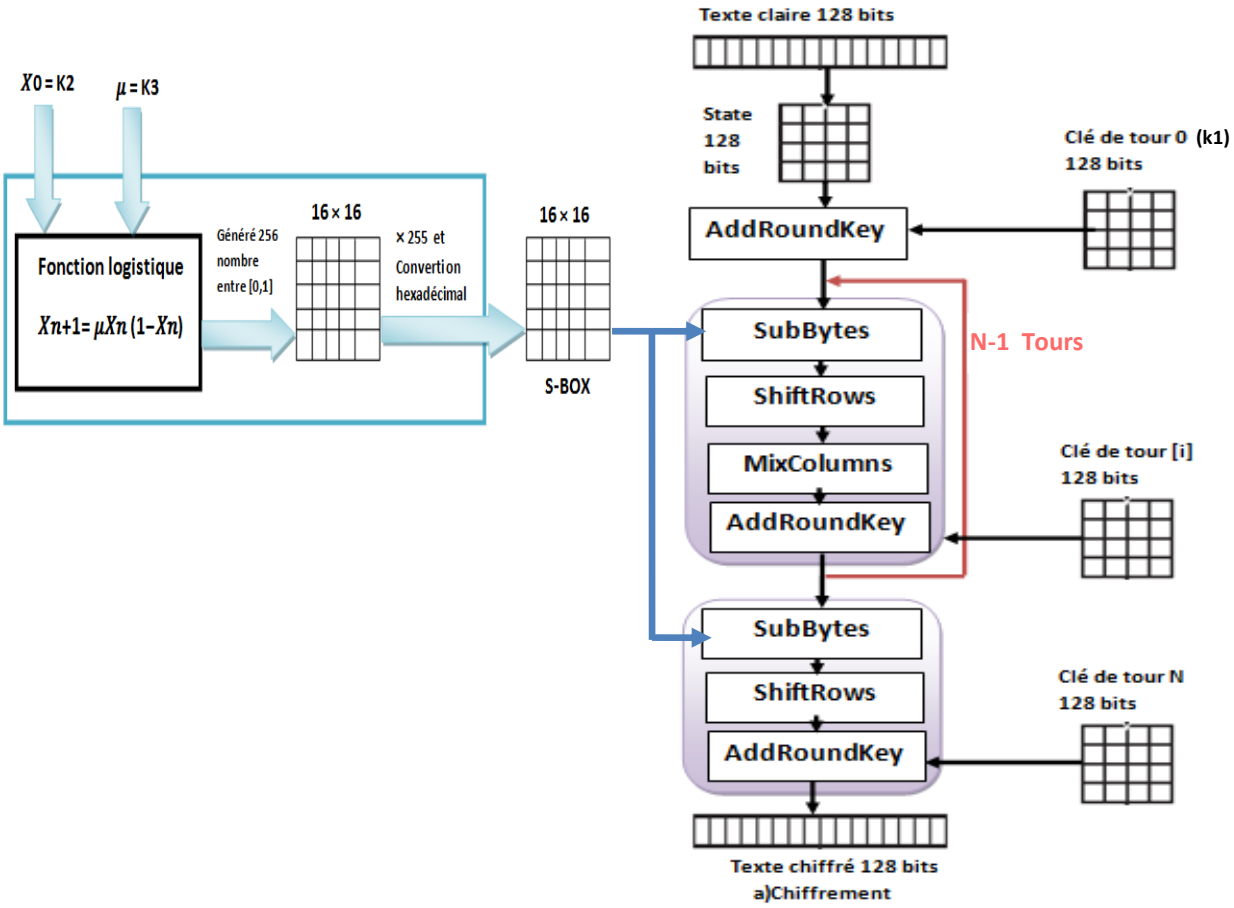

Figure 4.4 : chiffrement AES avec SBOX généré pseudo-aléatoirement.

Algorithme de chiffrement

1. Choisir un texte clair M de taille N bits.
2. Echange de clé à base de courbe elliptique :
   a. Choisir le type de courbe elliptique avec P point générateur.
   b. Choisir un grand nombre entier a.
   c. Calculer $Q_a = a*P$ et l'envoyer à l'entité B.
   d. Recevoir $Q_b = b*P$ de l'entité B.
   e. Calculer $K= a*Q_b =a*b*P$ qui représente la clé secrète établie entre l'entité A et B.
3. Extraction de trois clé K1, K2, K3 à partir de la clé K.
4. Génération pseudo-aléatoire de SBOX :
   a. Générer une matrice S de 16*16 de nombre pseudo-aléatoire entre [0, 1] avec la fonction logistique qui prend comme paramètre :
      - La valeur initiale X0 = K2.
      - Le paramètre de contrôle μ = K3.

b. Multiplier la matrice S par 255.

c. Convertir les résultats en hexadécimal et obtenir SBOX.

5. Chiffrement du texte M avec l'algorithme AES avec la clé K1 :
   a. AddRoundKey ().
   b. Répéter n-1 fois :
      - SubBytes(SBOX).
      - ShiftRows ().
      - MixColumns ().
      - AddRoundKey ().
   c. SubBytes(SBOX).
   d. ShiftRows ().
   e. AddRoundKey ().
6. Obtenir le texte chiffré C de taille N bits.

4.3.3.2 Déchiffrement

Pour le déchiffrement, après la génération de SBOX avec la fonction logistique on applique une fonction inverse pour obtenir INVSBOX, la clé K1 est utilisée dans le déchiffrement symétrique AES, comme l'indique la figure suivante :

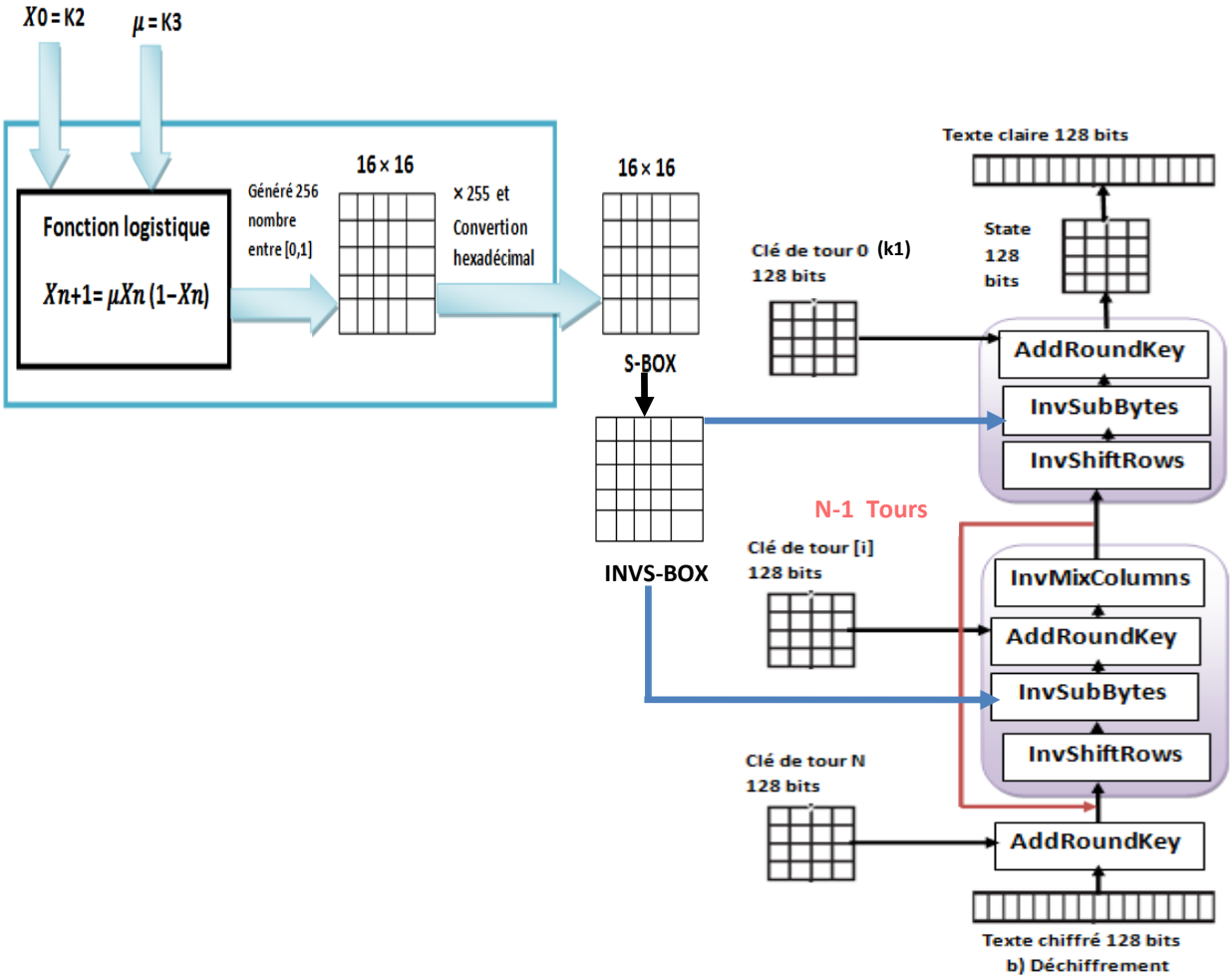

Figure 4.5 : déchiffrement AES avec INVSBOX généré pseudo-aléatoirement.

Algorithme de déchiffrement

1. Recevoir le texte chiffré C de taille N bits.
2. Echange de clé à base de courbe elliptique :
   a. Choisir le type de courbe elliptique avec P point générateur.
   b. Choisir un grand nombre entier b.
   c. Calculer $Q_b = b*P$ et l'envoyer a l'entité A.
   d. Recevoir $Q_a = a*P$ de l'entité A.
   e. Calculer $K= b*Q_a = b*a*P$ qui représente la clé secrète établie entre l'entité A et B.
   f. Extraction de trois clé K1, K2, K3 à partir de la clé K.
3. Génération pseudo-aléatoire d'INVSBOX :
   a. Générer une matrice S de 16*16 de nombre pseudo-aléatoire entre [0, 1] avec la fonction logistique qui prend comme paramètre :
      - La valeur initiale X0 = K2.

- Le paramètre de contrôle μ = K3.

b. Multiplier la matrice S par 255.

c. Convertir les résultats en hexadécimal et obtenir SBOX.

d. Appliquer une fonction inverse sur SBOX pur obtenir INVSBOX

4. Déchiffrement du texte C avec l'algorithme AES avec la clé K1 :

a. AddRoundKey ().

b. Répéter n-1 fois :

- InvShiftRows ().
- InvSubBytes(INVSBOX).
- AddRoundKey ().
- InvMixColumns ().

c. InvShiftRows ().

d. InvSubBytes(INVSBOX).

e. AddRoundKey ().

5. Obtenir le texte clair M de taille N bits.

Les problèmes de l'AES sont résolus de la manière suivante :

- ✓ Le problème d'échange de clé secrète et distribution des clés est résolue par l'utilisation de cryptosystème asymétrique ECC cryptographie à base des courbes elliptiques qu'on va la détailler dans la partie suivante.
- ✓ Le problème des tables statiques est résolu par la cryptographie chaotique avec l'utilisation de la fonction logistique pour générer SBOX INVSBOX de manière pseudo-aléatoire qu'on va la détailler dans le chapitre suivant.

On peut résumer toutes les étapes précédentes avec un schéma global dans la figure 4.6 suivante :

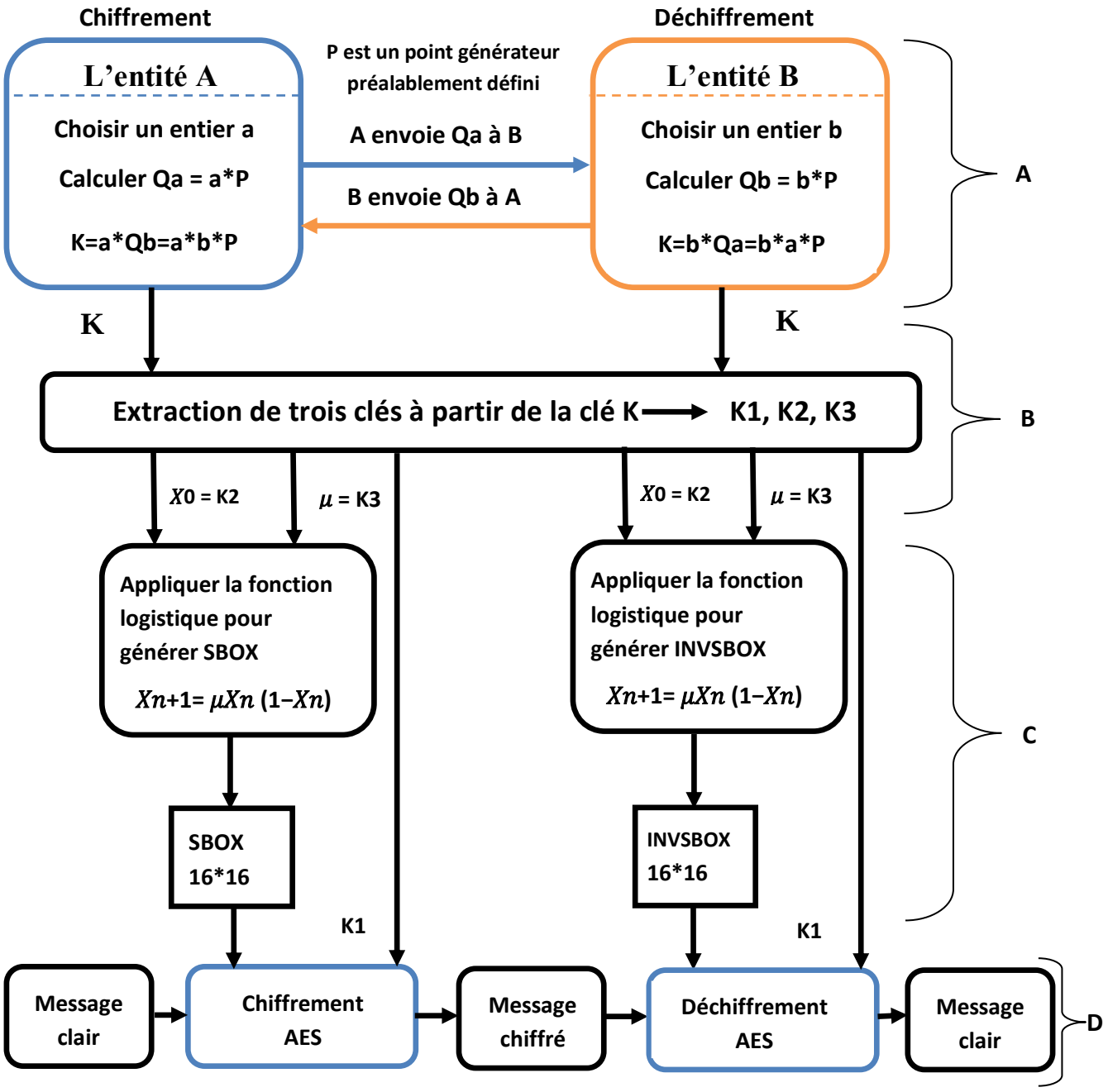

Figure 4.6 : Le cryptosystème hybride «chiffrement, déchiffrement ».

Dans la figure 4.6 :

A) Echange et établissement de la clé K entre les deux entités avec le protocole d'échange de clé de Diffie-Hellmann basé sur les courbes elliptiques.

B) Extraction de trois clés à partir de la clé K, k1 pour le chiffrement et le déchiffrement AES, k2 et k3 utilisé par la fonction logistique

C) en utilisant les clés K2, K3 dans la fonction logistique qui génère pseudo-aléatoirement SBOX et INVSBOX utilisées dans le chiffrement et le déchiffrement AES.

D) Processus de chiffrement utilise la clé K1 et SBOX généré, processus de déchiffrement utilise la clé K1 et INVSBOX généré.

4.4 Analyse comparative entre ECC et notre cryptosystème

Afin de montrer la rapidité de notre cryptosystème, Nous allons le comparer avec la méthode ECC en terme de temps d'exécution.

C'est pourquoi nous avons fait une implémentation de ECC et de notre cryptosystème avec trois entrées d'échantillons de 128 bits, 256 bits et 384 bits.

Nous avons choisir 3 types de courbes parmi les courbes proposées par la recommandation du NIST [39] sur GF(p) : {P - 160; P - 384; P – 521}.

Ces courbes sont illustrées dans les tableaux suivants, tel que :

- a, b et p : les coefficients de : $x^3+ax+b \bmod p$.
- r : Ordre de point de base.
- G(x, y) : Le point de base.

Selon la taille de la clé a, b, p, r seront classées comme suit :

Courbe P-160	
A	FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF 7FFFFFFC
P	FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF 7FFFFFFF
R	01 00000000 00000000 0001F4C8 F927AED3 CA752257
B	1C97BEFC 54BD7A8B 65ACF89F 81D4D4AD C565FA45
Gx	4A96B568 8EF57328 46646989 68C38BB9 13CBFC82
Gy	23A62855 3168947D 59DCC912 04235137 7AC5FB32

Table 4.1 : Courbe P-160.

Courbe P-384	
A	FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFC
P	FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFF
R	FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973
B	b3312fa7 e23ee7e4 988e056b e3f82d19 181d9c6e fe814112 0314088f 5013875a c656398d 8a2ed19d 2a85c8ed d3ec2aef
Gx	aa87ca22 be8b0537 8eb1c71e f320ad74 6e1d3b62 8ba79b98 59f741e0 82542a38 5502f25d bf55296c 3a545e38 72760ab7
Gy	3617de4a 96262c6f 5d9e98bf 9292dc29 f8f41dbd 289a147c e9da3113 b5f0b8c0 0a60b1ce 1d7e819d 7a431d7c 90ea0e5f

Table 4.2 : Courbe P-384.

Courbe P-521	
A	01FF FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFC
P	01FF FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFF FFFFFFFFFFFFFFFF
R	01FF FFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFF FFFFFFFA 51868783 BF2F966B 7FCC0148 F709A5D0 3BB5C9B8 899C47AE BB6FB71E 91386409
B	051 953eb961 8e1c9a1f 929a21a0 b68540ee a2da725b 99b315f3 b8b48991 8ef109e156193951 ec7e937b 1652c0bd 3bb1bf07 3573df88 3d2c34f1 ef451fd4 6b503f00
Gx	c6 858e06b7 0404e9cd 9e3ecb66 2395b442 9c648139 053fb521 f828af60 6b4d3dba a14b5e77 efe75928 fe1dc127 a2_a8de 3348b3c1 856a429b f97e7e31 c2e5bd66
Gy	118 39296a78 9a3bc004 5c8a5fb4 2c7d1bd9 98f54449 579b4468 17afbd17 273e662c 97ee7299 5ef42640 c550b901 3fad0761 353c7086 a272c240 88be9476 9fd16650

Table 4.3 : Courbe P-521.

Les expériences sont effectuées sur Netbeans 8.1 sur un processeur intel-core i5-4210U avec 4 GB de RAM sous la plate-forme Windows 7. L'expérimentation a été menée pour trouver le laps de temps lors du processus de chiffrement et le processus de déchiffrement par ECC et notre cryptosystème, L'efficacité de notre cryptosystème sur ECC est illustrée dans les tableaux suivants :

128 bits				
	ECC		**Notre cryptosystème**	
La taille de la clé	**Chiffrement (ms)**	**Déchiffrement (ms)**	**Chiffrement (ms)**	**Déchiffrement (ms)**
160/128	671	436	124	124
384/192	1295	608	250	250
521/256	1841	656	343	343

Table 4.4 : 128 bits- temps de chiffrement et de déchiffrement.

256 bits				
	ECC		Notre cryptosystème	
La taille de la clé	**Chiffrement (ms)**	**Déchiffrement (ms)**	**Chiffrement (ms)**	**Déchiffrement (ms)**
160/128	780	453	125	140
384/192	1669	640	250	265
521/256	2589	670	375	391

Table 3.5 : 256 bits- temps de chiffrement et de déchiffrement.

384 bits				
	ECC		Notre cryptosystème	
La taille de la clé	**Chiffrement (ms)**	**Déchiffrement (ms)**	**Chiffrement (ms)**	**Déchiffrement (ms)**
160/128	889	468	155	156
384/192	2262	655	266	281
521/256	3807	764	389	405

Table 3.6 : 384 bits- temps de chiffrement et de déchiffrement.

Nous allons voir les résultats obtenus de temps de chiffrement et déchiffrement dans les figures suivantes :

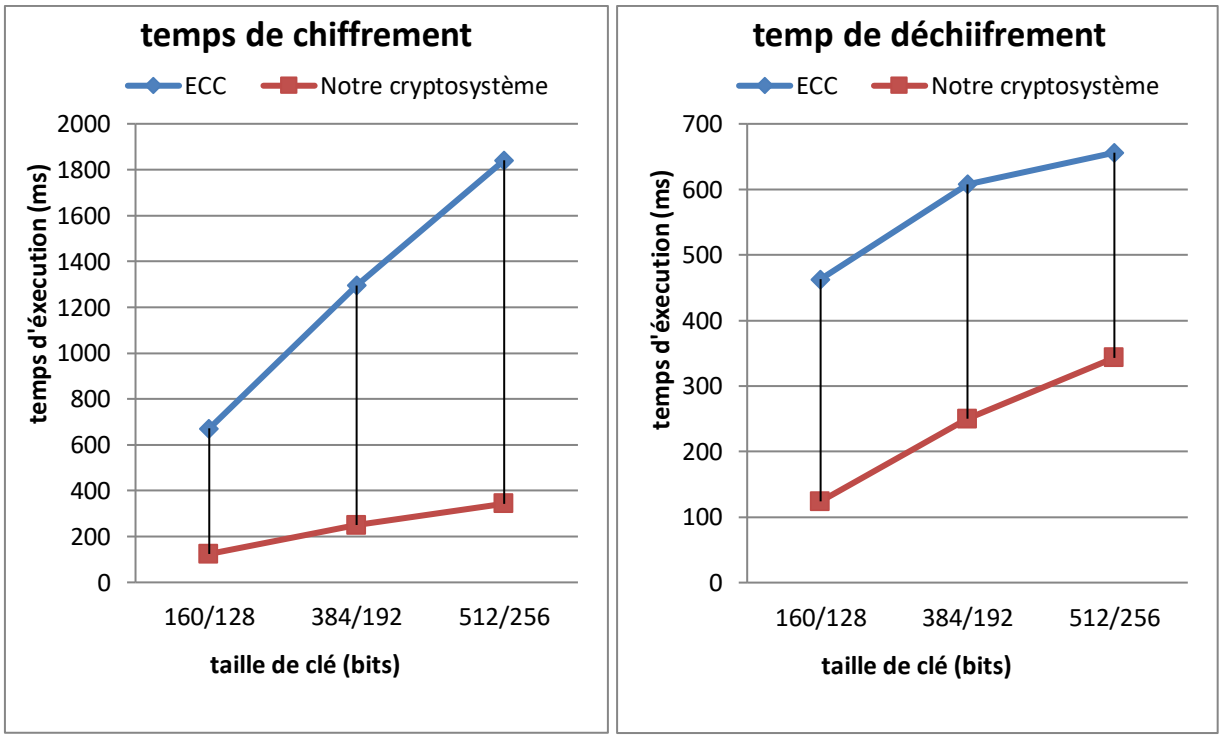

Figure 4.7 : comparaison entre ECC et notre cryptosystème128 bits.

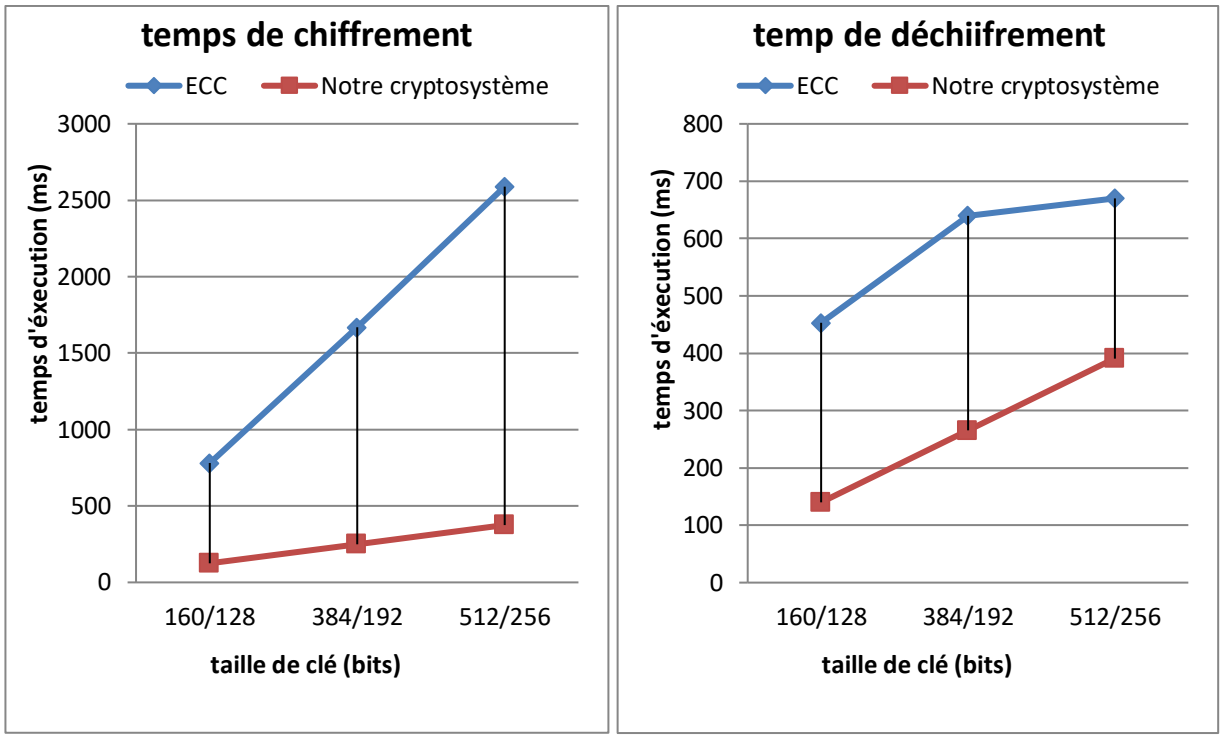

Figure 4.8 : comparaison entre ECC et notre cryptosystème256 bits.

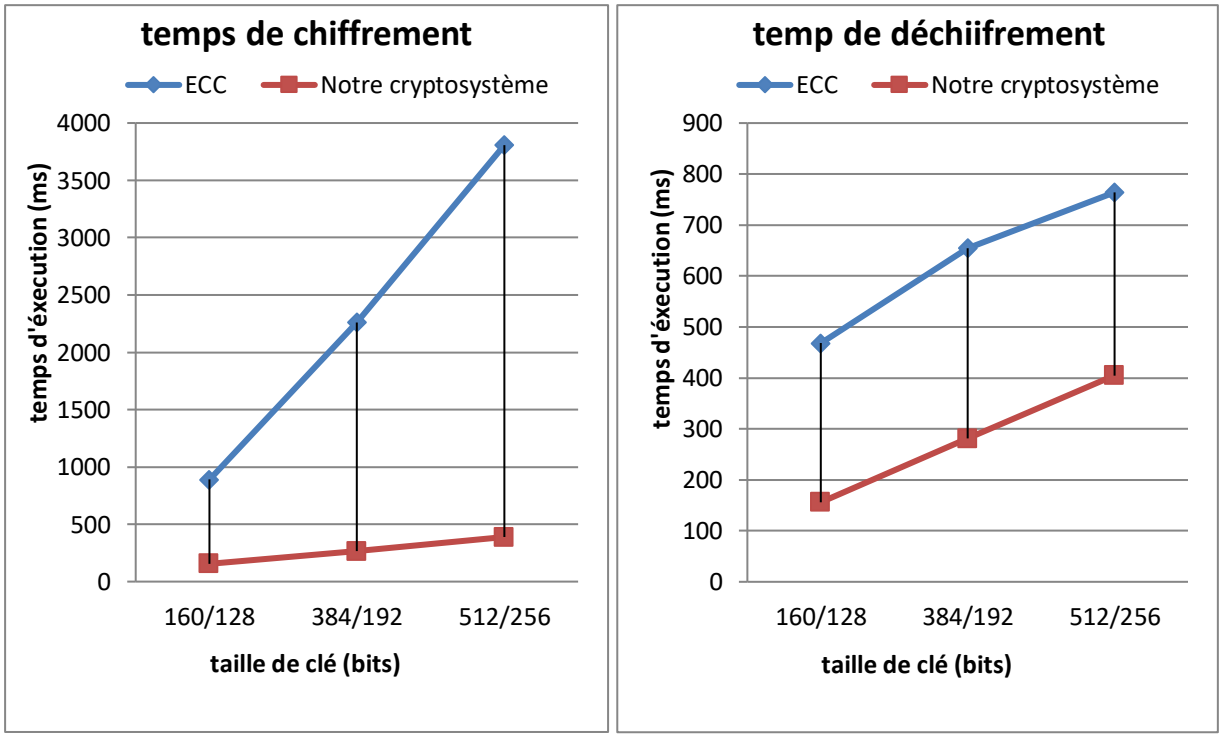

Figure 4.9 : comparaison entre ECC et notre cryptosystème 384 bits.

Notre cryptosystème hybride combine entre les deux méthodes de cryptographie qui sont AES et ECC afin de bénéficier de la rapidité et un bon niveau de sécurité.

AES fournit un bon niveau de sécurité et la rapidité de chiffrement et déchiffrement mais il a deux problèmes, l'un concernant l'utilisation des tables statiques SBOX et INVBOX qui est résolu par la fonction logistique et l'autre concernant l'échange de clé secrète qui est résolu par l'utilisation de la cryptographie à base des courbes elliptiques. Ces deux méthodes sont complémentaires et leur union procure un haut niveau de sécurité.

On est arrivé à un temps d'exécution raisonnable le comparant avec AES parce qu'on a ajouté l'échange de clé et à un temps d'exécution optimal le comparant avec ECC.

4.5 Conclusion

Dans ce chapitre, nous avons proposé un modèle de chiffrement hybride qui combine entre la rapidité de chiffrement symétrique en utilisant la méthode AES et la cryptographie à base des courbes elliptiques qui nous permet d'échanger les clés secrètes.

Dans le chapitre suivant, certains détails concernant la réalisation d'un cas d'application et les outils technologiques utilisés seront présentés.

Chapitre 05

Chapitre 05 : Application pour la gestion des comptes bancaires

5.1 Introduction

Après avoir présenté notre cryptosystème qui a donné de bons résultats en termes de temps d'exécution et de niveau de sécurité. Pour bien éclaircir l'efficacité de cryptosystème ; on va l'intégrer dans une application client / Serveur de gestion des comptes bancaires. Ce dernier, nous permet un échange sécurisé des données.

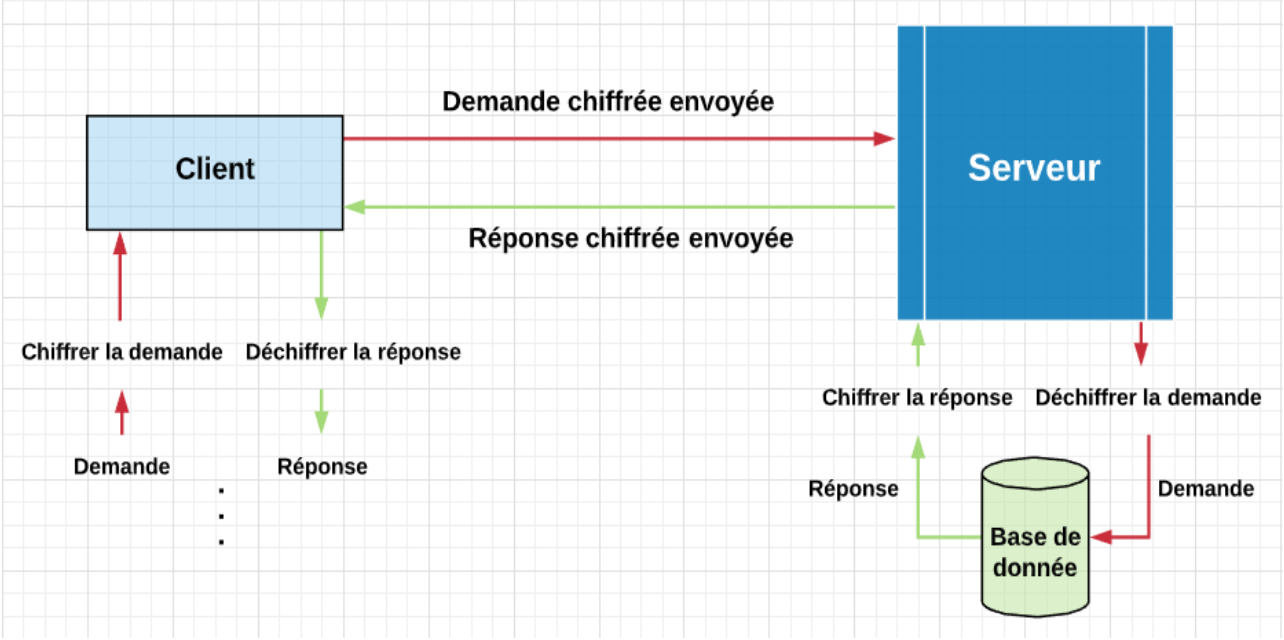

Figure 5.1 : Schéma intégration du cryptosystème dans une application client serveur.

5.2 Analyse et conception

5.2.1 Présentation UML

UML est un langage de modélisation graphique conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. C'est un support de communication performant qui permet grâce à sa représentation graphique, de concevoir des solutions, de faciliter la comparaison et de les évoluer. Il est couramment utilisé en développement logiciel et en conception orientée objet.

5.2.2 Diagramme de cas d'utilisation

Il est utilisé pour donner une vision globale du comportement fonctionnel d'un système logiciel. Un cas d'utilisation représente une unité discrète d'interaction entre un utilisateur et un système. Il est une unité significative de travail. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs, ils interagissent avec les cas d'utilisation.

Un acteur participe au moins à un cas d'utilisation qui peut être un utilisateur humain ou un autre système. Ce dernier interagit directement avec le système étudié.

Dans notre application de gestion des comptes bancaires, nous avons trois acteurs qui sont :

1. Client : D'abord, le client fait un échange de clé cryptographique avec le serveur ensuite il accède au système via un contrôle d'accès (login et mot de passe). Puis il peut effectuer l'une ou plusieurs parmi les opérations suivantes :

- Consulter solde.
- Transfer d'argent.
- Changer mot de passe.
- Afficher la liste des transactions.

2. Employé : Premièrement, il fait un échange de clé cryptographique avec le serveur ensuite il accède au système via un contrôle d'accès (login et mot de passe). Enfin il peut effectuer l'une ou plusieurs parmi les opérations suivantes :

Créer compte client.

- Dépôt (versement) d'argent.
- Retrait d'argent.
- Transfert d'argent.
- Affichage de la liste des comptes.

3. Administrateur : Au premier lieu, il échange la clé cryptographique avec le serveur ensuite il accède au système via un contrôle d'accès (login et mot de passe). Enfin il peut effectuer l'une ou plusieurs parmi les opérations suivantes :

- Créer compte employé
- Afficher la liste des employés.
- Supprimer compte employer.
- Modifier compte employer.
- Afficher la liste des comptes client.

- Modifier compte client.
- Supprimer compte client.
- Afficher la liste des clients.
- Afficher la liste des transactions.
- Afficher la liste des opérations.

5.2.2.1 Diagramme de cas d'utilisation Client

Le diagramme de cas d'utilisation client représente les interactions du client avec le système comme nous les illustrons dans le digramme suivant :

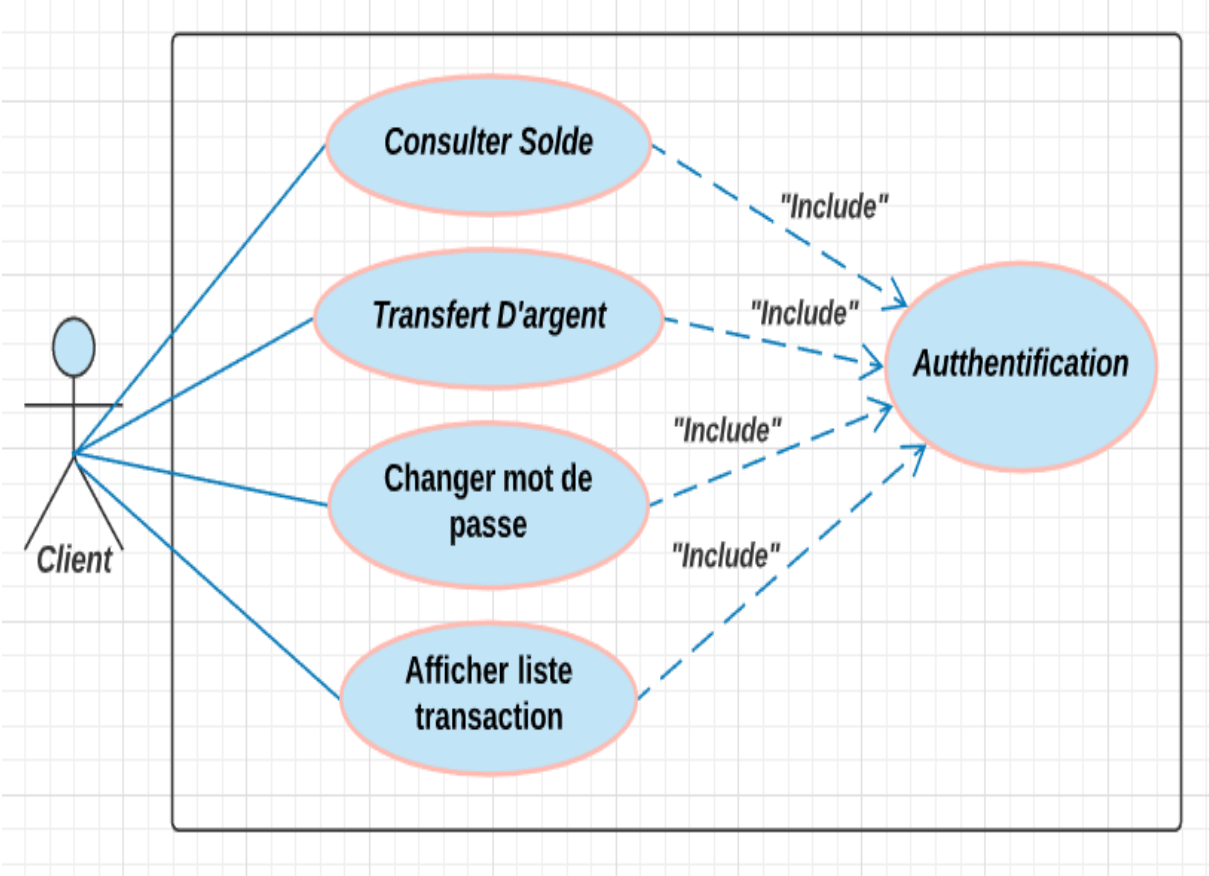

Figure 5.2 : diagramme de cas d'utilisation client.

5.2.2.2 Diagramme de cas d'utilisation employé

Le diagramme de cas d'utilisation employé représente les interactions d'employé avec le système comme nous les illustrons dans le digramme suivant :

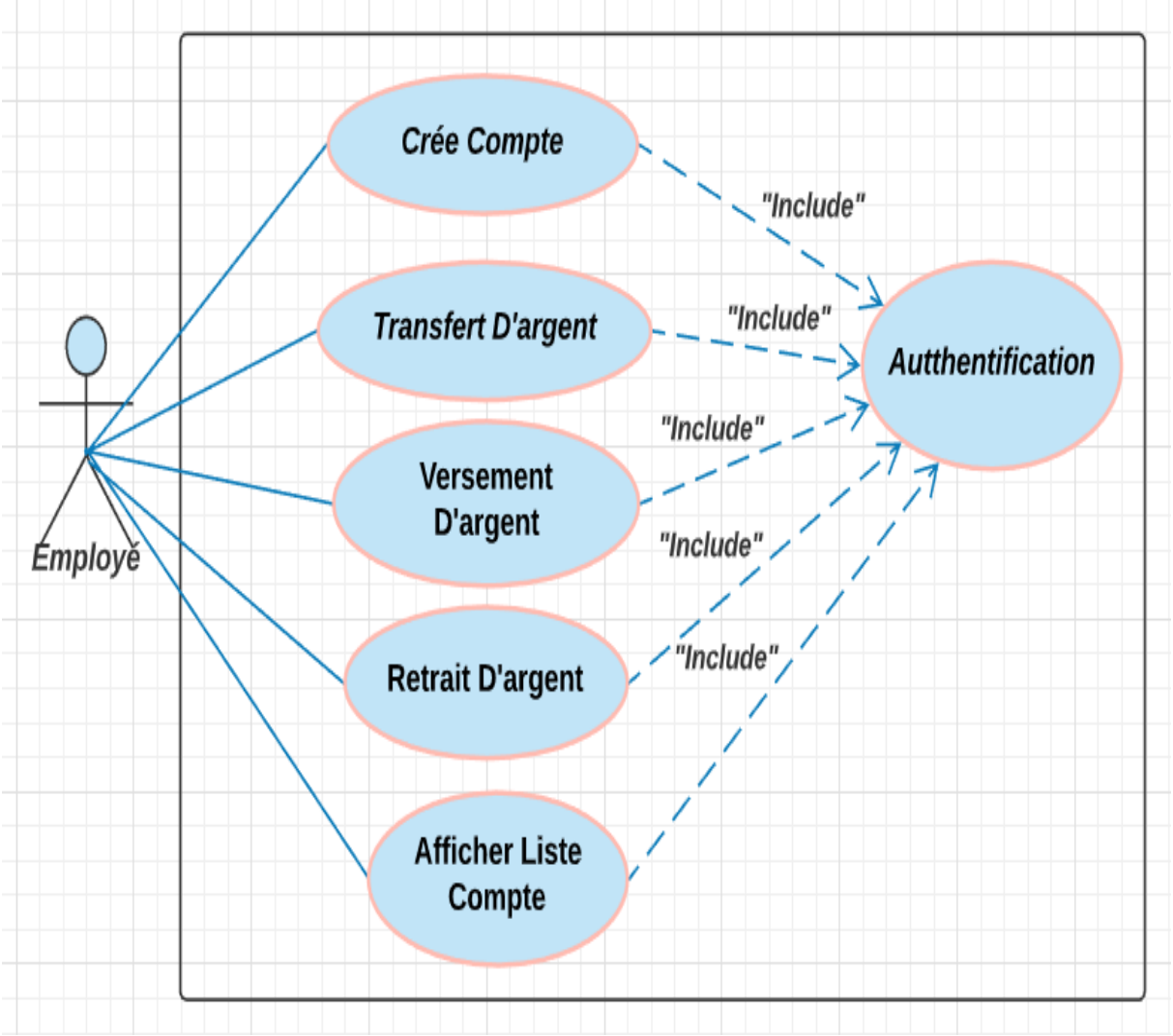

Figure 5.3 : diagramme de cas d'utilisation employé.

5.2.2.3 Diagramme de cas d'utilisation Administrateur

Le diagramme de cas d'utilisation Administrateur représente les interactions d'administrateur avec le système comme nous les illustrons dans le digramme suivant :

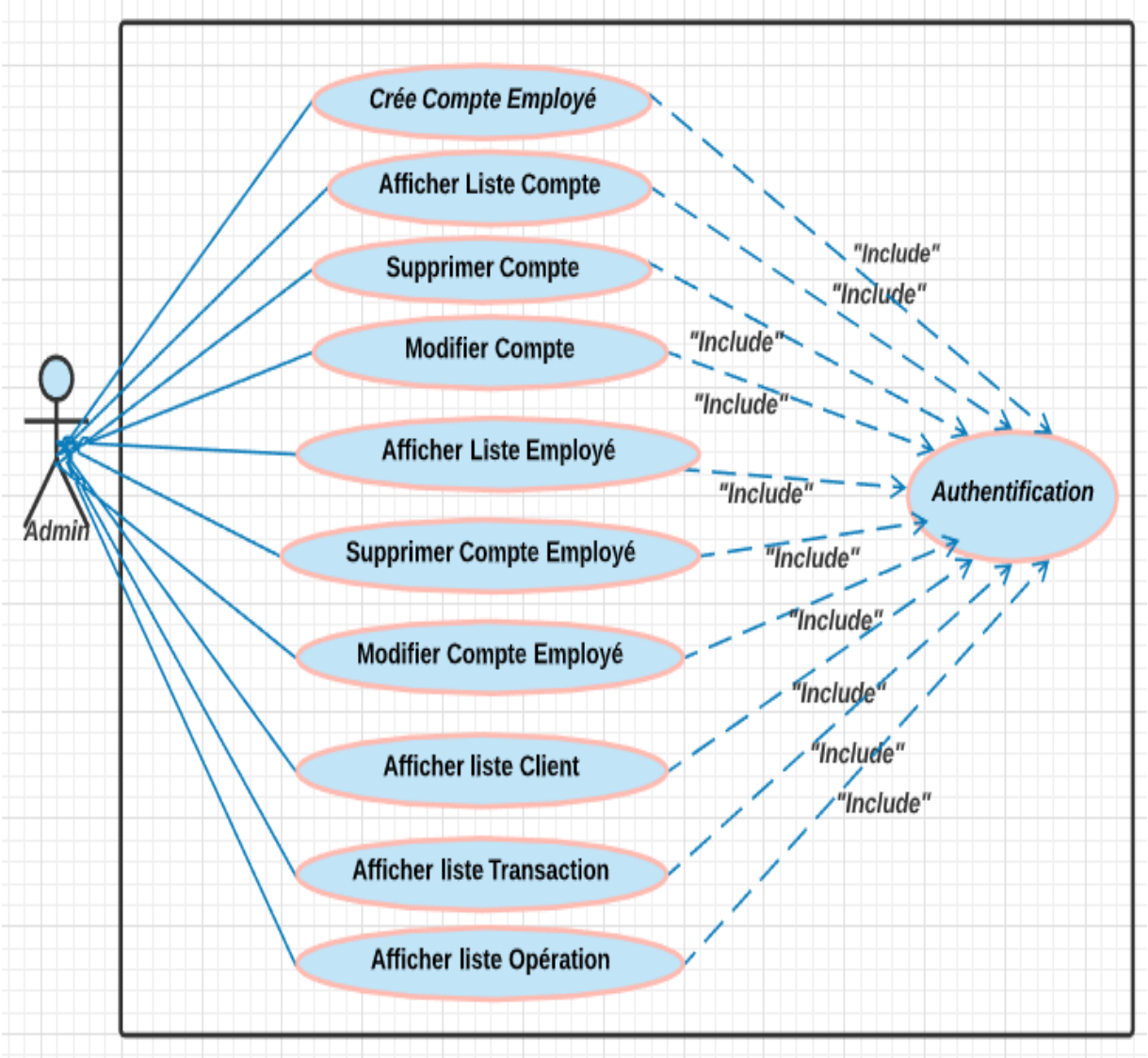

Figure 5.4 : diagramme de cas d'utilisation Administrateur.

5.2.2 Diagramme de classe

Ce digramme permet de donner la représentation statique du système à développer. Cette représentation est centrée sur les concepts de classe et d'association. Chaque classe se décrit par les attributs et les opérations.

Un digramme de classe se définit comme un ensemble de classes contenant des attributs et des opérations, reliées les unes aux autres par des relations et ceci ayant des conditions de participation (cardinalités).

Une classe décrit un groupe d'objets ayant les mêmes propriétés (attributs), un même comportement (opérations) et une sémantique commune. Un attribut est une propriété élémentaire d'une classe qui prend une valeur. Une opération est une fonction applicable aux objets d'une classe. Elle est également appelée méthode.