الجمهورية الجزائرية الديمقراطية الشعبية
République algérienne démocratique et populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique
جامعة عين تموشنت بلحاج بوشعيب
Université –Ain Temouchent- Belhadj Bouchaib
Faculté des Sciences et de Technologie
Département des Mathématiques et Informatique

Projet de Fin d'Etudes
Pour l'obtention du diplôme de Master en : Informatique
Spécialité : Cyber Sécurité et Intelligence Artificielle

Thème

EXPLORATION DU CHIFFREMENT DE MERKLE-HELLMAN BASÉ SUR LE PROBLÈME DU SAC À DOS À L'AIDE DE COURBES ELLIPTIQUES

Présenté par :

1) Melle Yousra Meguenni
2) Melle Narimane Dada

Devant le jury composé de :

M. F. BENARIBI	UAT.B.B (Ain Temouchent)	Président
M. A. BENZERBADJ	UAT.B.B (Ain Temouchent)	Examinateur
M. H. BOUCHAKOUR ERRAHMANI	UAT.B.B (Ain Temouchent)	Encadreur

Année Universitaire 2023/2024

Dédicace

Nous tenons à exprimer notre profonde gratitude et notre reconnaissance à tous ceux qui ont contribué à notre parcours et à notre réussite. A nous meme, dont le soutien et la collaboration ont été inestimables tout au long de cette aventure. Ensemble, nous avons surmonté les défis et célébré les succès.nous dédions ce modest travail à nos parents, pour leur amour inconditionnel, leur soutien sans faille et leurs encouragements constants. Sans vous, rien de tout cela n'aurait été possible et bien sur a nos fréres et soeurs qui sont toujours avec nous.

À nos ingénieurs et encadreurs de stage chez Ooredoo et Djezzy, merci pour votre patience, votre expertise et vos conseils précieux. Votre encadrement a été crucial pour notre développement professionnel et personnel. À ceux qui nous ont aidés à trouver ce stage, merci pour votre soutien et vos efforts. Vous avez ouvert des portes et créé des opportunités qui ont été essentielles à notre croissance.

Remerciement

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui nous voudrions témoigner notre reconnaissance. Nous voudrions tout d'abord adresser toute notre gratitude à notre encadrant, monsieur Hichem BOUCHAKOUR ERRAHMANI, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter notre réflexion. Merci pour votre inestimable support.

Nous voudrions exprimer notre reconnaissance envers nos enseignants du département de mathématiques et d'informatique qui nous ont apporté leur support moral et intellectuel tout au long de notre cursus. Un grand merci aux membres du jury qui nous ont honorés en acceptant de juger ce modeste travail.

Nous désirons également remercier les collègues de la promotion CYSIA et RID qui nous ont apporté leur support moral, leur confiance et leur collaboration tout au long de ces années. À nos amis, pour leur présence, leur soutien et les moments de détente et de joie qui nous ont permis de garder notre équilibre.

Résumé

Ce mémoire explore une amélioration potentielle du chiffrement de Merkle-Hellman, un algorithme de chiffrement asymétrique basé sur le problème du sac à dos (knapsack problem), en intégrant les courbes elliptiques. Le problème du sac à dos est connu pour sa complexité computationnelle, mais les progrès en cryptanalyse ont mis en évidence certaines faiblesses du chiffrement de Merkle-Hellman classique. Les courbes elliptiques, en revanche, offrent une sécurité renforcée avec des clés de taille plus petite grâce à leurs propriétés mathématiques uniques.

Dans ce travail, nous étudions les pré-requis d'un tel cryptosystème avec les différentes bases mathématiques et cryptographiques, comme nous analysons quelques approches qui adoptent le problème de sac à dos basées sur les courbes elliptiques, où nous apprenons l'existence des différents méthodes conçues.

En combinant ces deux techniques, nous visons à créer un système de chiffrement plus robuste contre les attaques modernes. Les résultats montrent que cette approche hybride améliore significativement la sécurité tout en augmentant la complexité computationnelle, soulignant la nécessité d'un équilibre entre sécurité et efficacité.

Mots Clés : Courbe Elliptique, Chiffrement de Merkel-Hellman, Problème de sac à dos, Cryptographie Asymétrique

Abstract

This thesis explores a potential improvement of the Merkle-Hellman encryption, an asymmetric encryption algorithm based on the knapsack problem, by integrating elliptic curves. The knapsack problem is known for its computational complexity, but advances in cryptanalysis have highlighted certain weaknesses in the classical Merkle-Hellman encryption. Elliptic curves, on the other hand, offer enhanced security with smaller key sizes due to their unique mathematical properties.

In this work, we study the prerequisites of such a cryptosystem, including various mathematical and cryptographic foundations. We also analyze some approaches that adopt the knapsack problem based on elliptic curves, where we learn about the existence of different designed methods.

By combining these two techniques, we aim to create a more robust encryption system against modern attacks. The results show that this hybrid approach significantly improves security while increasing computational complexity, highlighting the need for a balance between security and efficiency.

Keywords : Elliptic Curve, Merkle-Hellman Encryption, Knapsack Problem, Asymmetric Cryptography

Sommaire

Table des figures

Liste des tableaux

LIST OF ALGORITHMS

Exploration du Chiffrement de Merkle-Hellman basé sur le Problème du Sac à Dos à l'aide de Courbes Elliptiques

Narimane Dada et Yousra Meguenni

June 2024

CHAPITRE 1

INTRODUCTION GÉNÉRALE

Le chiffrement est un domaine crucial dans la cryptographie moderne, essentiel pour assurer la sécurité des informations sensibles. Parmi les nombreuses techniques de chiffrement, l'algorithme de Merkle-Hellman basé sur le problème du sac à dos (ou "knapsack problem") a suscité un intérêt particulier depuis son introduction. Ce sujet se situe à l'intersection de la cryptographie, des mathématiques appliquées et de la théorie des algorithmes. L'environnement de notre étude est donc celui de la sécurité informatique et des systèmes cryptographiques, avec un focus sur les cryptosystèmes basés sur des problèmes mathématiques complexes.

Le problème du sac à dos, en tant que problème NP-complet, offre une base théorique robuste pour la construction de systèmes cryptographiques sécurisés. Cependant, les versions initiales du chiffrement Merkle-Hellman ont été compromises, ce qui a mis en évidence la nécessité de renforcer ces méthodes. La problématique principale de cette recherche est donc de déterminer comment améliorer la sécurité du chiffrement Merkle-Hellman en intégrant les propriétés des courbes elliptiques, qui sont largement reconnues pour leur efficacité et leur sécurité dans le domaine de la cryptographie. Pour répondre à cette problématique, notre recherche propose une nouvelle approche combinant le chiffrement de Merkle-Hellman avec les courbes elliptiques. Nous visons à démontrer que cette combinaison peut offrir une sécurité renforcée tout en maintenant une efficacité pratique pour le chiffrement et le déchiffrement. Notre démarche inclut une analyse théorique approfondie de l'algorithme proposé, suivie d'une implémentation pratique et d'une évaluation de sa performance et de sa sécurité.

Ce mémoire tend ainsi à démontrer que le chiffrement de Merkle Hellman cryptographiques en utilisant les courbes elliptiques est une technique fiable qui permet de maintenir le principe initial de n'importe quel cryptosystème, le chiffrement de merkle hellman nous fournit une solution à la sécurité des messages et les courbes elliptique fortifie de plus en plus cette notion, en ajoutant une complexité aux calculs.L'étude du chiffrement de merkle-hellman permet en effet de connaitre les techniques de générer les cléset chiffrement/déchiffrement. Après un chapitre consacré aux courbes elliptiques, où l'on apprend les différentes notions mathématiques nécessaires sur les opérations de bases dans les courbes elliptiques. Afin de suivre l'évolution des travaux dans le domaine du sujet, on a consacré un chapitre sur l'état de l'art de où on étudie quelques articles. Enfin, on verra dans un dernier chapitre sur notre contribution dans le chiffrement de

merkle-hellman en utilisant les courbes elliptique, tout en tâchant de respecter les points étudiés.

CHAPITRE 2

CHIFFREMENT DE MERKLE-HELLMAN

Sommaire

2.1 Introduction

La cryptologie, englobant à la fois la cryptographie et la cryptanalyse, est une discipline essentielle pour la protection des informations dans notre monde numérique. La cryptographie se concentre sur la création de systèmes sécurisés pour protéger les données, tandis que la cryptanalyse vise à découvrir des failles potentielles dans ces systèmes.

Dans ce chapitre, nous allons explorer le cryptosystème de Merkle-Hellman, un des premiers algorithmes de chiffrement asymétrique basé sur le problème combinatoire du sac à dos. Nous commencerons par des notions de base en cryptologie, en clarifiant les concepts de cryptographie et de cryptanalyse, ainsi que les différences entre les clés symétriques et asymétriques. Nous introduirons ensuite des notions mathématiques fondamentales telles que l'inverse modulaire et le calcul modulo. Ces concepts nous mèneront à unecompréhension approfondie du cryptosystème de Merkle-Hellman, de la génération des clés jusqu'aux processus de chiffrement et de déchiffrement.

2.2 Concepts de base

2.2.1 Cryptologie

La cryptologie est la science du secret qui existe depuis l'antiquité. Elle fait partie d'une science plus vaste dite la sécurité informatique. Elle se constitue de deux grandes branches ,la cryptographie et la cryptanalyse, que nous allons définir au cours de cette partie [3].

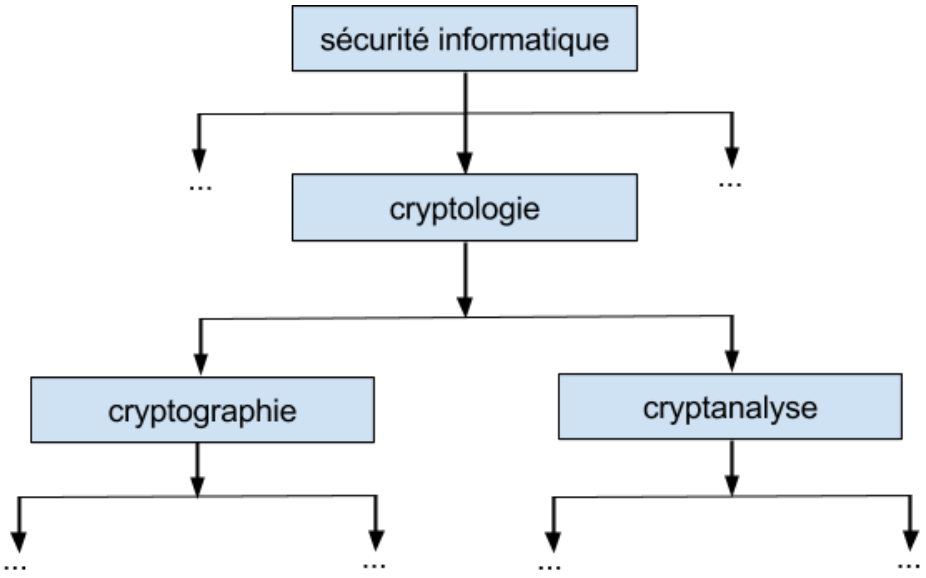

FIGURE 2.1 – Les branches de la cryptologie

2.2.2 Cryptographie

La cryptographie est le processus de masquage ou de codage des informations afin que seule la personne à laquelle un message était destiné puisse les lire. L'art de la cryptographie est utilisé pour coder les messages depuis des milliers d'années et continue d'être utilisé dans les cartes bancaires, les mots de passe informatiques et le commerce électronique.

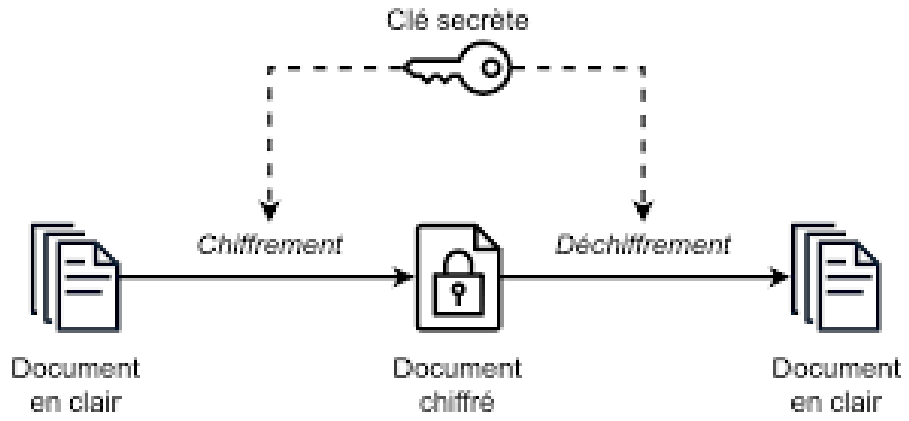

FIGURE 2.2 – La définition de cryptographie

La cryptographie a pour but :

— **La confidentialité :** La confidentialité est un service utilisé pour garder le contenu de l'information de tous, sauf ceux autorisés à l'avoir. Il existe de nombreuses approches pour assurer la confidentialité, allant de la protection physique aux algorithmes mathématiques qui rendent les données inintelligibles.

— **L'intégrité des données :** L'intégrité des données est un service qui traite de la modification non autorisée des données. Pour assurer l'intégrité des données, il faut pouvoir détecter la manipulation de données par des parties non autorisées. La manipulation de données inclut des éléments tels que l'insertion, la suppression et la substitution.

— **L'authentification :** L'authentification est un service lié à l'identification. Cette fonction s'applique aux entités et à l'information elle-même. Deux parties qui entrent dans une communication doivent s'identifier. Les informations fournies sur une chaîne doivent être authentifiées quant à leur origine, leur date d'origine, leur contenu, leur temps d'envoi, etc. Pour ces raisons, cet aspect de la cryptographie est habituellement subdivisé en deux grandes classes : authentification d'entité et authentification d'origine de données. L'authentification d'origine de données fournit implicitement l'intégrité des données (si un message est modifié, la source a changé).

— **La non-répudiation :** La non-répudiation est un service qui empêche une entité de nier des engagements ou des actions antérieurs. Lorsque des conflits surviennent en raison d'une entité refusant que certaines actions aient été prises, un moyen de résoudre la situation est nécessaire. Par exemple, une entité peut autoriser l'achat de biens par une autre entité et plus tard nier qu'une telle autorisation a été accordée.

Une procédure impliquant un tiers de confiance est nécessaire pour résoudre le différend [10].

Un objectif fondamental de la cryptographie est de traiter adéquatement ces quatre domaines en théorie et en pratique. La cryptographie concerne la prévention et la détection de la triche et d'autres activités malveillantes.

En cryptographie, il existe plusieurs mécanismes pour chiffrer l'information, où chacun a sa propre utilité. Dans cette partie, nous allons définir les différentes formes de chiffrement qui existent.

Symétrique

La cryptographie à clé symétrique, également appelée cryptage à clé unique, est une technique de cryptage qui repose sur une clé secrète unique pour le cryptage et le décryptage des données.

Dans cette méthode, l'expéditeur et le destinataire doivent utiliser exactement la même clé secrète pour comprendre les données. Elle consiste à transformer des données normales en code secret (texte chiffré) à l'aide de la clé secrète et d'un processus mathématique spécifique.

Lorsque le destinataire, qui connaît également la clé secrète, reçoit le message codé secrètement, il peut utiliser le même processus mathématique pour le retransformer en données normales. De cette manière, le destinataire obtient l'information originale à partir du code secret.

FIGURE 2.3 – Le Mécanisme de chiffrement à clef privée(Symétrique)

Asymétrique

La cryptographie asymétrique utilise des paires de clés – une clé privée gardée secrète et une clé publique partagée ouvertement. Vous pouvez utiliser la clé publique d'une personne pour crypter un message, et elle seule peut le décrypter à l'aide de sa clé privée. Cette méthode renforce la sécurité numérique en permettant une communication sûre sans partage des clés secrètes, ce qui est essentiel dans notre monde en ligne.

Cette méthode est utile pour communiquer en toute sécurité, car les destinataires n'ont besoin que de votre clé publique. Elle élimine le risque de partager une clé symétrique secrète. Ces « algorithmes à clé publique » utilisent une paire de clés pour sécuriser les données [2].

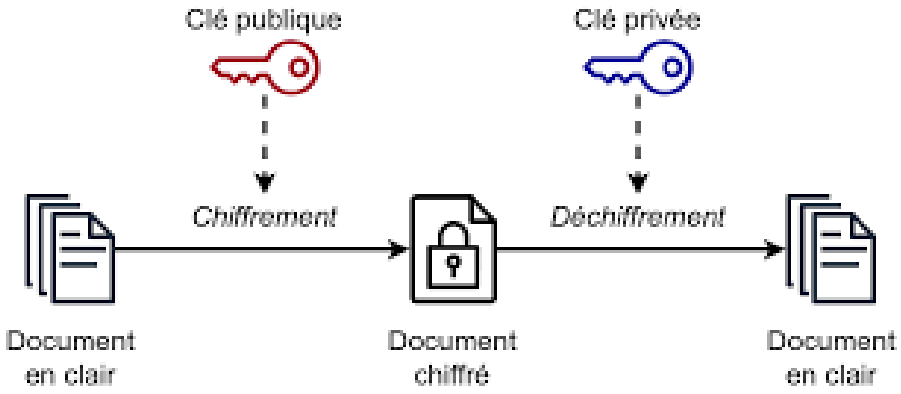

FIGURE 2.4 – Le Mécanisme de chiffrement à clef publique

2.2.3 Cryptanalyse

La cryptanalyse est la branche de la cryptologie concernée par la résolution des systèmes cryptographiques utilisés par d'autres sans connaissance de la clé, inventée par les Arabes dans l'âge d'or de la civilisation islamique. Les objectifs des cryptanalystes (spécialistes de la cryptanalyse) sont destinés à lire le texte des messages cryptés et à récupérer les systèmes cryptographiques utilisés. Le texte est récupéré pour sa valeur de renseignement potentielle. En revanche, les systèmes sont récupérés pour être appliqués à des messages futurs dans des systèmes identiques ou similaires.

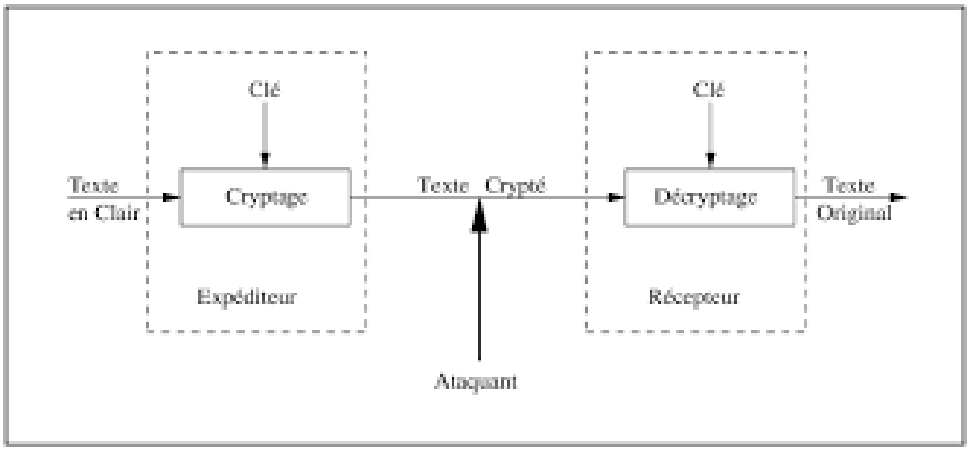

FIGURE 2.5 – La définition de cryptanalyse

Les principales attaques qui peuvent être utilisées par les cryptanalystes sont (liste non exhaustive) :

— L'attaque sur un texte chiffré connu : où le cryptanalyste ne connaît que les textes chiffrés pour essayer de retrouver la clé.

— L'attaque sur un texte clair connu : où le cryptanalyste connaît des textes clairs ainsi que leurs versions chiffrées, pour essayer de trouver des informations secrètes comme la clé de chiffrement.

— L'attaque sur un texte clair choisi : où le cryptanalyste peut choisir des textes en clair spéciaux qui offriront plus d'informations sur la clé [3].

2.3 Notion Mathématique

La cryptographie moderne s'inspire beaucoup de la mathématique, car elle se base sur ces opérations et fonctions, ce qui nous oblige de définir quelques opérations mathématiques utilisées dans la cryptographie. Les plus importantes sont :

2.3.1 Le modulo

L'opérateur modulo, noté (mod), est souvent utilisé en mathématique ainsi qu'en informatique pour désigner le reste de la division euclidienne, il est équivalent à cette équation :

$$x \mod y = x - \left(\left\lfloor \frac{x}{y} \right\rfloor \cdot y \right)$$

Exemple

$$11 \mod 5 = 11 - \left(\left\lfloor \frac{11}{5} \right\rfloor \cdot 5 \right) = 11 - 2 \cdot 5 = 1.$$

2.3.2 L'inverse modulaire

En mathématiques, et plus précisément en arithmétique modulaire, l'inverse modulaire de l'entier a modulo n est un entier u satisfaisant l'équation :

$$a \cdot u \equiv 1 \pmod{n}.$$

On calcule généralement l'inverse de a (noté aussi a^{-1}) avec l'algorithme d'Euclide. La définition est équivalente à :

$$u \equiv a^{-1} \pmod{n}.$$

L'inverse de a modulo n existe si et seulement si a et n sont premiers entre eux (c'est-à-dire si $\mathrm{PGCD}(a, n) = 1$). Si cet inverse existe, l'opération de division par a modulo n équivaut à la multiplication par son inverse.

2.3.3 La puissance modulair

La puissance modulaire ou bien l'exponentiation modulaire est une élévation à la puissance opérée modulo un nombre entier. Elle est largement utilisée dans le domaine de la cryptologie, et elle s'exprime sous la forme suivante :

$$C \equiv b^e \pmod{n},$$

où b, e, n sont des entiers, nommés respectivement base, exposant et module. C est le résultat et appartient à l'ensemble des entiers finis $[0, n[$. Plusieurs méthodes existent pour calculer la puissance modulaire, la plus connue est l'algorithme de l'exponentiation rapide. Exemple :

$$5^2 \mod 13 = 12.$$

2.4 Problème du sac à dos

C'est un problème qui est très simple. Supposons une pile d'objets, chacun avec un poids différent, est-il possible de mettre certains de ces objets dans un sac à dos tout en sachant que ce dernier a un poids maximum ? Plus formellement, supposons qu'on a une série de valeurs $P_1, P_2, \ldots, P_n$ et une somme S, calculez les valeurs des $b_1, b_2, \ldots, b_n$ tels que :

$$S = b_1 \cdot P_1 + b_2 \cdot P_2 + \ldots + b_n \cdot P_n.$$

Les valeurs de b_i peuvent être des 0 ou des 1. Le "un" indique que l'objet est dans le sac et le "zéro" indique qu'il n'y est pas. Par exemple, les objets doivent peser respectivement 1, 5, 6, 11, 14 et 20. On ne peut remplir qu'un sac à dos qui pèse 22 en utilisant les poids 5, 6 et 11. Par contre, on ne pourra pas faire de même avec un sac qui pèse 24. En général, le temps requis pour résoudre ce genre de problème semble croître exponentiellement avec le nombre d'objets dans la pile. L'idée soutenue par l'algorithme de Merkle-Hellman est d'encoder un message comme une solution à une série de problèmes de sac à dos. Un bloc de texte en clair de longueur égale au nombre d'objets dans la pile sélectionnera les objets dans le "sac à dos" (le texte en clair correspondant à la valeur de b), et le texte chiffré serait la somme résultante.

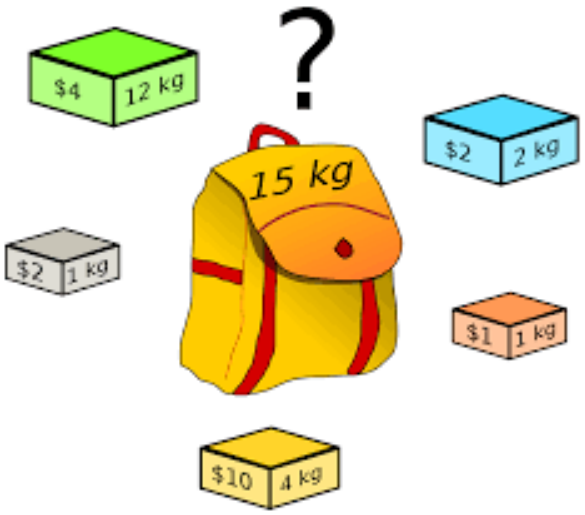

FIGURE 2.6 – Problème de sac à dos (Le problème du sac à dos : quelles boîtes choisir afin de maximiser la somme emportée tout en ne dépassant pas les 15 kg autorisés ?)

2.4.1 Les empilements

Il existe deux types de problèmes d'empilement :

— Une soluble en temps linéaire,

— Une soluble en temps exponentiel.

Empilement facile

Si la liste des poids est super-croissante, on utilise un algorithme (appelé glouton cf. annexe A) de la manière suivante :

1. Prendre le poids total (il est à signaler que le poids total ici n'est pas la somme de tous les éléments dans la suite) et le comparer avec le plus grand nombre de la suite.
   — Si le poids total est inférieur à ce nombre, alors celui-ci n'est pas dans le tas. On recommence l'opération avec le nombre suivant dans le tas (qui, par définition de la suite, sera plus petit).
   — Si le poids total est supérieur à ce nombre, alors celui-ci est dans le tas.
2. Réduire le poids du tas à créer de ce nombre et passer au plus grand nombre suivant de la suite.
3. Répéter jusqu'à ce que ce soit terminé.
4. Si le poids total a pu être ramené à 0 : il y a une solution.

Empilement difficile

Dans le cas présent, on ne connaît pas d'algorithme rapide. Il faut tester méthodiquement toutes les solutions possibles, ce qui, si la suite des poids est suffisamment longue, est impraticable. Ces algorithmes sont en temps exponentiels.

Le crypto système de Merkle-Hellman exploite cette propriété. La clé privée est une suite de poids super-croissante. À partir de celle-ci, on calcule la clé publique. Ce calcul consiste à prendre la suite super-croissante, et à la multiplier par $n \mod m$, avec m supérieur à la somme de tous les termes de la suite, et n ne devant avoir aucun facteur commun avec m.

Remarque : L'empilement facile peut être transformé pour créer un empilement difficile. Pour la clé publique, on utilisera un empilement difficile qui servira à chiffrer. La clé privée quant à elle, utilisera un empilement facile, qui donne un moyen simple de déchiffrer les messages. Bien sûr, ceux qui ne connaissent pas la clé privée sont obligés de résoudre le problème d'empilement difficile, ce qui est infaisable en pratique [4].

2.5 Crypto système de Merkle-Hellman

2.5.1 Historique

Le premier cryptosystème asymétrique, aménagé par les deux scientifiques Merkle et Hellman en 1978, est basé sur le problème de la somme de sous-ensemble (un cas particulier du problème de sac à dos). Dans ce procédé, si les éléments de la suite des poids sont super-croissants (sac facile), alors on peut résoudre le problème en temps polynomial avec un algorithme glouton. Sinon (sac dur), la seule manière de résolution de ce problème est de faire un essai exhaustif examinant toutes les solutions possibles, ce qui est impraticable avec une suite très longue. C'est pour cela que la clé publique n'est pas une suite super-croissante, contrairement à la clé privée. Au cours des premières années de ce cryptosystème, les cryptologues pensaient que ce dernier était l'avenir de la cryptologie, mais malheureusement, six ans après, Adi Shamir a cassé ce cryptosystème et l'a prouvé comme vulnérable.

2.5.2 Définition de Séquence super-croissante

Soit une suite u_n et la série associée $S_n = \sum_{i=1}^{n} u_n$. On dit que u_n est super-croissante si l'on a :

$$\forall n,\ u_{n+1} > S_n$$

Notons tout de suite que les bases de numérations sont toutes des exemples de séquence super-croissante ($1 + 2 < 4$, $1 + 2 + 4 < 8$, etc).

Il est clair que le problème du sac à dos est trivial lorsque l'on utilise une séquence super-croissante. Il est tout aussi clair qu'il n'a pas toujours de solution.

Ainsi, si l'on choisit la base 2 comme séquence super-croissante, tout nombre pourra se décomposer sur cette base au sens du problème du sac à dos. Il n'en va pas de même si l'on choisit la base 10 (par exemple, $13 = 8 + 4 + 1$ se décompose aisément sur la séquence $\{1, 2, 4, 8\}$, alors que le même nombre ne peut pas se décomposer sur la séquence $\{1, 10\}$).

L'idée de Merkle-Hellman est de transformer un problème du sac à dos trivial basé sur une séquence super-croissante en problème du sac à dos général, qui devient alors NP-complet, donc "insoluble". Nous allons tout d'abord présenter une version simplifiée de l'algorithme Merkle-Hellman.

2.5.3 Algorithme glouton

Soit un sac $S = \{o_1, \ldots, o_n\}$ et k le poids visé. On va décomposer k en soustrayant le plus grand o_i possible à chaque fois jusqu'à obtenir 0. Les o_i ainsi utilisés nous donnent la solution au problème.

2.5.4 Génération des clés

Les étapes nécessaires pour générer les clés de chiffrement et de déchiffrement de notre fameux algorithme sont :

1. Tout d'abord, une séquence super-croissante de n éléments $W = \{w_1, w_2, \ldots, w_n\}$ est créée (de préférence longue), c'est la base d'une clé privée. De là, calculez la somme ;
2. Ensuite, choisissez un nombre Q supérieur à la somme ;
3. De plus, choisissez un nombre R inclus dans l'intervalle $[1, Q[$ et premier avec Q ;
4. La clé privée se compose de Q, W et R ;
5. Pour calculer une clé publique, générez la séquence B en multipliant chaque élément de W par $R \mod Q$, $B_i = W_i \times R \mod Q$;
6. La séquence B constitue la clé publique.

Exemple : Soit la suite super-croissante suivante $W = \{2, 5, 8, 18, 38, 77\}$, constituée de $n = 6$ éléments. La somme des éléments de W est 148.

— On prend $Q = 158$ par exemple ;

— On choisit un nombre R premier avec Q, soit ce nombre 139 ;

— La clé privée est : $\{\{2, 5, 8, 18, 38, 77\}, Q, R\}$

Calculons la clé publique :

$$2 \times 139 \mod 158 = 120$$

$$5 \times 139 \mod 158 = 63$$

$$8 \times 139 \mod 158 = 6$$

$$18 \times 139 \mod 158 = 132$$

$$38 \times 139 \mod 158 = 68$$

$$77 \times 139 \mod 158 = 117$$

Donc la clé publique est : $B = \{120, 63, 6, 132, 68, 117\}$.

2.5.5 Chiffrement

Pour chiffrer un message M en utilisant le cryptosystème de Merkle-Hellman, il faut d'abord le convertir en binaire, et si le message est trop long, il est coupé en blocs de n bits au plus. En utilisant la clé publique B, on chiffre le message binaire en calculant :

$$C = \sum_{i=1}^{n} M_i \times B_i.$$

Exemple : En utilisant les clés décrites précédemment, on veut chiffrer le message binaire suivant $m = 100110$. On calcule la somme suivante :

$$1 \times 120 + 0 \times 63 + 0 \times 6 + 1 \times 132 + 1 \times 68 + 0 \times 117 = 320.$$

Le message chiffré est : 320.

2.5.6 Déchiffrement

Pour déchiffrer un message, le récepteur calcule :

$$P = R^{-1} \times C \pmod{Q},$$

d'abord il calcule $R^{-1} \pmod{Q}$ avec l'algorithme d'Euclide étendu. Maintenant, il résout l'instance

$$P = \sum_{i=1}^{n} X_i \cdot W_i,$$

où X_i peut prendre 1 ou 0, en utilisant un algorithme glouton. Il décompose P en sélectionnant le W_i le plus grand qui est inférieur ou égal à P. Puis il recommence jusqu'à obtenir 0. Les éléments choisis de la suite privée W donnent le message initial, autrement dit, les X_i sont le message en clair calculé.

Exemple : Continuons avec le même exemple précédent, donc on veut déchiffrer le message chiffré $C = 320$. D'abord, on calcule l'inverse de R avec l'algorithme d'Euclide

étendu, on trouve $R^{-1} = 133$. Ensuite, on résout

$$P = R^{-1} \times C \pmod{Q}.$$

En appliquant les valeurs que nous avons, on trouve

$$P = 133 \times 320 \pmod{158} \Rightarrow P = 42560 \pmod{158} \Rightarrow P = 58.$$

Maintenant, on calcule

$$P = \sum_{i=1}^{n} X_i \cdot W_i$$

avec l'algorithme glouton :

$$58 - 38 = 20$$

$$20 - 18 = 2$$

$$2 - 2 = 0.$$

Les éléments tirés du sac sont 38, 18, 2, qui ont pour places dans la clé privée respectivement 1, 4, 5, d'où X_i vaut 1 pour ces éléments, et 0 pour les autres, c'est-à-dire : $X = \{1, 0, 0, 1, 1, 0\}$, ce qui est le message clair m[1].

2.5.7 Cassage du cryptosystème de Merkle-Hellman

En 1982, Shamir découvre une attaque permettant d'obtenir le message clair en ne se servant que de la clé publique et du chiffre issu du cryptosystème de Merkle-Hellman, en utilisant l'algorithme de la réduction des réseaux.

2.5.8 Réduction de réseau

Pour attaquer le cryptosystème de Merkle et Hellman, on utilise un algorithme appelé réduction de réseau. Cet algorithme, inspiré de l'orthogonalisation classique de Gram-Schmidt, a des performances souvent meilleures que prévu, bien que son comportement soit encore mal compris.

Un réseau est un ensemble de points espacés de manière régulière. Mathématiquement, est défini comme une combinaison linéaire à coefficients entiers des vecteurs d'une base spécifique de ce réseau.. Cet ensemble est appelé "réseau engendré par la base $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n$" et est noté L. il peut être simplement défini par :

$$L = \mathbb{Z} \times \mathbf{b}_1 + \mathbb{Z} \times \mathbf{b}_2 + \ldots + \mathbb{Z} \times \mathbf{b}_n = \left\{ \sum_{i=1}^{n} r_i \times \mathbf{b}_i : (r_1, r_2, \ldots, r_n) \in \mathbb{Z}^n \right\}$$

Une base d'un réseau est un ensemble de vecteurs linéairement indépendants qui forment ce réseau. L'idée de la réduction de réseau est de calculer une nouvelle base qui engendre le même réseau.$(\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_n)$ mais dont les vecteurs sont plus courts et "plus orthogonaux" [4].

2.6 Conclusion

Le cryptosystème de Merkle-Hellman illustre une application ingénieuse des principes de la cryptographie asymétrique et des mathématiques combinatoires pour sécuriser les communications. En exploitant la complexité du problème de sac à dos, ce système propose une méthode de chiffrement robuste, bien que certaines vulnérabilités aient été découvertes au fil du temps, soulignant l'importance de la cryptanalyse dans l'évaluation de la sécurité des algorithmes cryptographiques. La génération des clés, ainsi que les processus de chiffrement et de déchiffrement, mettent en lumière les défis et les solutions propres à la cryptographie moderne. En rétrospective, le cryptosystème de Merkle-Hellman a non seulement contribué à l'évolution des techniques de cryptographie asymétrique mais a également pavé la voie pour des algorithmes plus avancés, nous rappelant l'importance continue de l'innovation et de l'analyse rigoureuse dans ce domaine en constante évolution.

CHAPITRE 3

LES COURBES ELLIPTIQUES

Sommaire

3.1 Introduction

La contrainte majeure rencontrée dans la conception des cryptosystèmes est le temps d'exécution face à la complexité des opérations utilisées. Le problème de la taille des clés représente également un problème pour les concepteurs, car ils optaient pour des clés de taille considérable afin d'augmenter le niveau de sécurité de leurs systèmes, ce qui rend les opérations beaucoup plus lourdes. Par conséquent, elles prennent plus de temps de calcul et exigent un matériel plus sophistiqué.

L'une des techniques qui ont révolutionné la sécurité informatique en apportant des solutions à ces contraintes sont les courbes elliptiques, qui commencent à être de plus en plus utilisées dans les cryptosystèmes modernes, car elles offrent des avantages de performances, que ce soit du point de vue de la taille des clés, du temps de calcul ou même de la complexité des opérations.

Dans ce chapitre, nous allons d'abord définir ce qu'est une courbe elliptique en présentant ses différentes caractéristiques. Nous entamerons par la suite les principales opérations nécessaires à la conception d'un cryptosystème sécurisé, ainsi que le problème du logarithme discret elliptique qui représente la base sur laquelle reposent ces systèmes. Enfin, nous aborderons la notion d'échange de clé par les courbes elliptiques, qui nous sera de grande utilité pour une bonne compréhension des différentes approches étudiées.

3.2 Définition des courbes elliptiques

Une courbe elliptique est l'ensemble des solutions à une équation de la forme

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

tel que les constantes A et B doivent vérifier l'équation $4A^3 + 27B^2 \neq 0$. Cette condition (appelée discriminant Δ) force le polynôme cubique à avoir des racines distinctes [5].

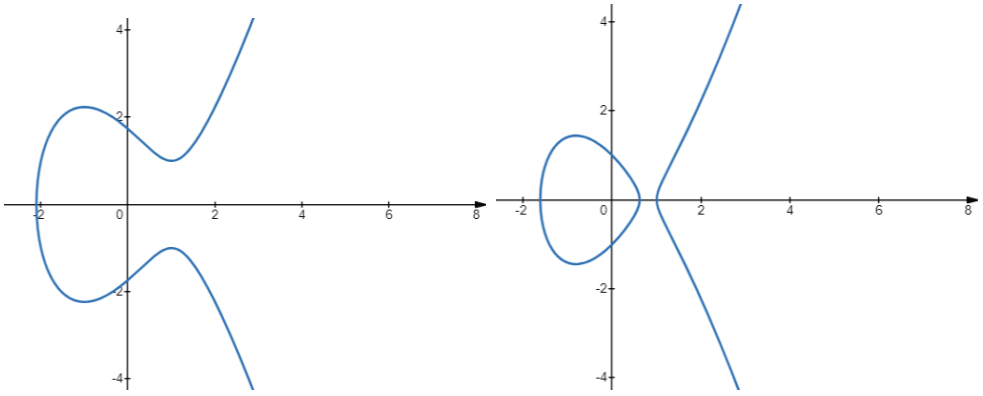

FIGURE 3.1 – Exemples de courbes elliptiques

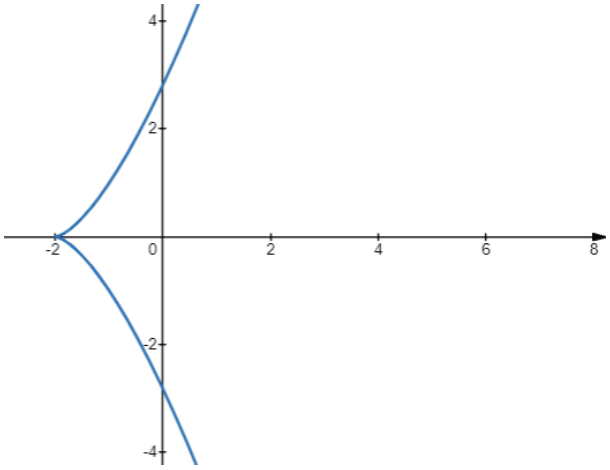

FIGURE 3.2 – Exemple d'une courbe non elliptique ($\Delta = 0$)

3.3 Loi de Composition sur les Courbes Elliptiques

3.3.1 L'inverse d'un point

Étant donné que les solutions de l'équation d'une courbe elliptique sont des racines carrées, leurs tracés ont une forme symétrique par rapport à l'axe des abscisses, tel que pour chaque valeur de x on trouve deux solutions $y = \sqrt{x^3 + ax + b}$ et $y = -\sqrt{x^3 + ax + b}$. Par conséquent, l'inverse d'un point sur la courbe n'est autre que son image symétrique par rapport à l'axe des abscisses. En considérant un point $P(x_p, y_p)$, son inverse $-P$ est défini par les coordonnées $-P(x_p, -y_p)$[7].

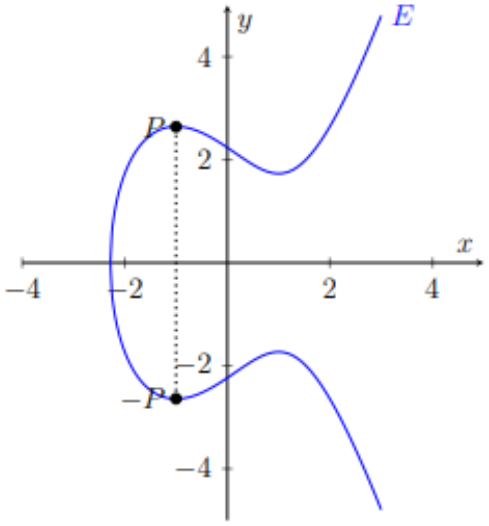

FIGURE 3.3 – L'inverse d'un point

3.3.2 Addition de deux points

Soient E une courbe elliptique définie sur un corps K, et deux points $P, Q \in E(K)$. Soit L la droite reliant P à Q (la tangente à E si $P = Q$) et R le troisième point d'intersection de L avec E. Soit L' la droite verticale passant par R. On définit $P+Q \in E(K)$ comme étant le deuxième point d'intersection de L' avec E. Muni de cette loi de composition, $(E(K), +)$ est un groupe abélien dont l'élément neutre est le point à l'infini $\mathcal{O}$ [7].

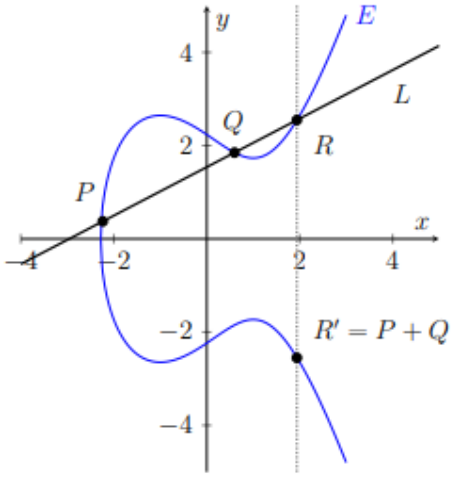

FIGURE 3.4 – Addition de deux points (courbe elliptique)

3.3.3 Double d'un point

Si P_1 et P_2 sont symétriques par rapport à l'axe des abscisses, on considère alors que la somme des deux points est nulle. Il s'agit donc du point à l'infini. Si la droite (L) est tangente en un point de la courbe, on considère qu'elle intersecte deux fois la courbe en ce point et par conséquent la somme de P_1 et P_2 (ou bien le doublement de P) est le symétrique du point de tangence [7].

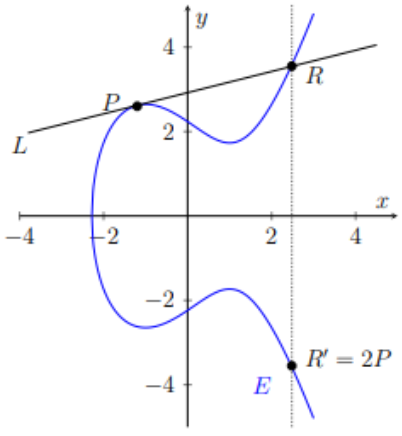

FIGURE 3.5 – Doublement d'un point sur une courbe elliptique

3.3.4 Multiple d'un point

La multiplication d'un point par un scalaire n est définie comme une suite d'opérations d'addition sur ce point :

$$n \cdot P = \underbrace{P + P + P + \cdots + P}_{n \text{ fois}} \tag{3.1}$$

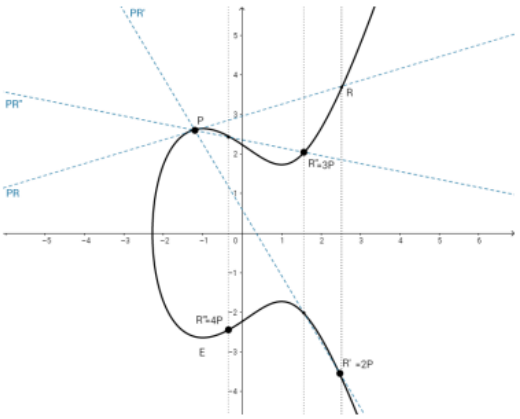

FIGURE 3.6 – Multiple d'un point

3.3.5 L'élément neutre

En outre le cas du double d'un point pour l'opération d'addition, nous pourrions rencontrer un deuxième cas particulier, il s'agit de l'addition d'un point P(xp, yp) avec son inverse P'(xp, yp). La ligne L traversant P et P' devient une ligne verticale d'équation x = xp, cette droite ne croise pas E dans un troisième point R. afin de déterminer l'addition dans ce cas, un point O définit à l'infini est présenté, tel qu'on prétend son existence sur chaque ligne verticale du plan, ce qui donne P + P'= $\mathcal{O}$ [7].

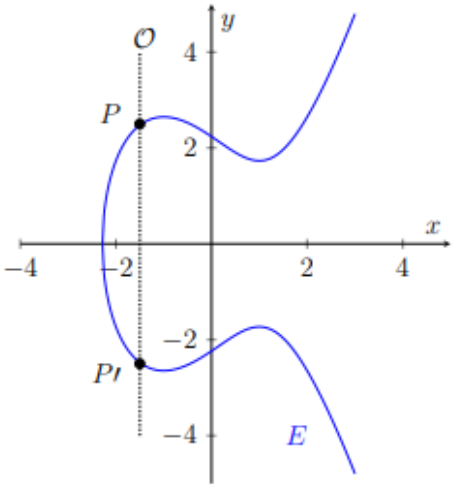

FIGURE 3.7 – l'élément neutre

3.4 Algorithme d'addition et double d'un point

Après avoir détaillé les différentes opérations pour une courbe elliptique comme notions mathématiques, on se trouve dans l'obligation de trouver un algorithme efficace qui nous permette de les implémenter pour nos cryptosystèmes. L'algorithme synthétisant ces opérations, de l'anglais *Double and Add*, est présenté comme suit :

Soit E une courbe elliptique définie par $E : y^2 = x^3 + Ax + B$, $4A^3 + 27B^2 \neq 0$. Et soient $P(x_p, y_p)$ et $Q(x_q, y_q)$ deux points de E.

Afin de définir les coordonnées de $R(x_r, y_r)$, tel que $R = -(P + Q)$, l'équation suivante est donnée :

$$x_r = \lambda^2 - x_p - x_q \quad \text{et} \quad y_r = \lambda(x_p - x_r) - y_p$$

tel que :

$$\lambda = \begin{cases} \frac{y_q - y_p}{x_q - x_p}, & \text{si } P \neq Q \\ \frac{3x_p^2 + A}{2y_p}, & \text{sinon} \end{cases}$$

Il suffit ensuite d'inverser l'ordonnée y_r en $-y_r$ pour définir $R' = -R = P + Q$ [2].

Algorithm 1 Addition et Double d'un point

Entrées : $P(x_p, y_p), Q(x_q, y_q)$
Sorties : $R(x_r, y_r)$

Algorithme
if $P == -Q$ **then**
    Retourner $\mathcal{O}$
else if $P == \mathcal{O}$ **then**
    Retourner Q
else
    if $P \neq Q$ **then**
        $\lambda \leftarrow \frac{(y_q - y_p)}{(x_q - x_p)}$
    else
        $\lambda \leftarrow \frac{(3 \times x_p^2 + A)}{(2 \times y_p)}$
    end if
    $x_r \leftarrow \lambda^2 - x_p - x_q$
    $y_r \leftarrow \lambda \times (x_p - x_r) - y_p$
    Retourner R
end if

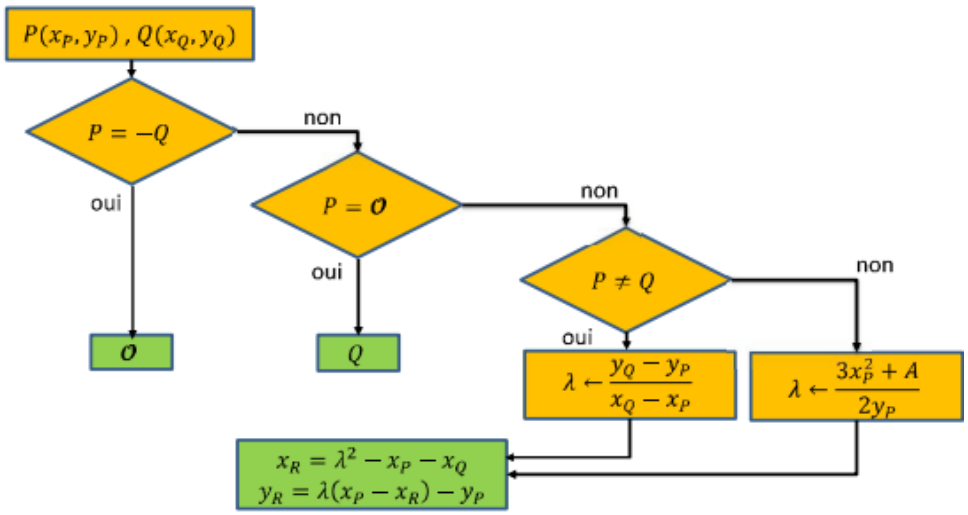

FIGURE 3.8 – Algorithme Addition et Double d'un point

3.5 Notion de groupe

Comme détaillé précédemment, pour n'importe quels deux points d'une courbe elliptique, l'opération d'addition donne toujours un point de la même courbe, ce qui fait de cette opération une loi de composition interne sur l'ensemble des points E qui résolvent son équation. À partir des précédentes définitions sur les opérations dans une courbe elliptique, nous pouvons synthétiser les propriétés suivantes [8] :

1. **Loi interne** : Si $P, Q \in E$ et $P + Q = R$, alors $R \in E$.
2. **Associativité** : Pour tout $P, Q, R \in E$, $(P + Q) + R = P + (Q + R)$ (voir Figure 9).
3. **Élément neutre** : L'élément à l'infini O est un élément neutre tel que $\forall P(x, y) \in E, P + O = P$.
4. **L'inverse d'un élément** : Chaque élément dans E admet un inverse tel que $\forall P(x, y) \in E, \exists - P(x, -y) \in E$.
5. **Commutativité** : L'addition de deux points de la courbe est une opération commutative, car la droite joignant P et Q croisant E dans R est la même droite joignant Q et P (l'ordre des points n'a pas d'importance), $\forall P, Q \in E$, $P + Q = Q + P$.

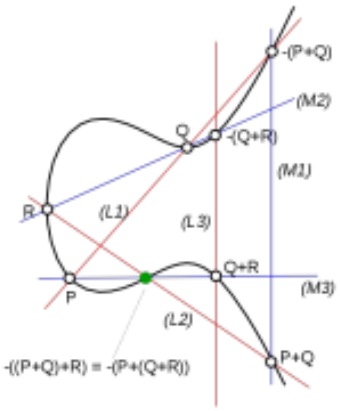

FIGURE 3.9 – Associativité de l'addition sur une courbe elliptique

D'après les propriétés suscitées, nous déduisons que (E, +) forme un groupe abélien (commutatif).[6]

3.6 Les courbes elliptiques sur les corps finis

Les exemples de courbes présentés précédemment sont définis dans l'ensemble des réels $\mathbb{R}$. En revanche, l'application des opérations sur les courbes elliptiques peut avoir lieu sur un corps fini [1], et c'est même exigé en cryptographie, où on implémente des courbes elliptiques définies sur un certain corps $\mathbb{F}_p$, tel que p est premier [**ref15**]. Donc, la définition d'une courbe elliptique cryptographique est donnée comme suit :

Une courbe elliptique sur un corps fini $\mathbb{F}_p$ avec p premier, est l'ensemble des coordonnées $X, Y \in \mathbb{F}_p$ qui résolvent l'équation

$$Y^2 \equiv X^3 + AX + B \pmod{p}$$

union O, tel que $A, B \in \mathbb{F}_p$ et $4A^3 + 27B^2 \not\equiv 0 \pmod{p}$.

Afin de définir les coordonnées du résultat $P + Q$, on complète les équations de calcul précédentes comme suit :

$$x_r = \lambda^2 - x_p - x_q \pmod{p} \quad \text{et} \quad y_r = \lambda(x_p - x_r) - y_p \pmod{p}$$

tel que :

$$\lambda = \begin{cases} \frac{y_q - y_p}{x_q - x_p} \pmod{p}, & \text{si } P \neq Q \\ \frac{3x_p^2 + A}{2y_p} \pmod{p}, & \text{sinon} \end{cases}$$

et pour définir l'ordonnée de l'inverse de R, $-y_r \pmod{p}$.

Exemple 1 : Courbe elliptique sur un corps fini Soit la courbe $E : y^2 = x^3 + 3x + 8$. Le graphe de E sur l'ensemble des réels $\mathbb{R}$ est représenté par la Figure 10.

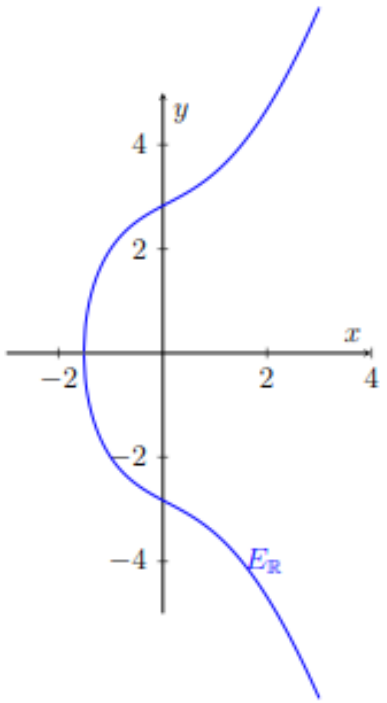

FIGURE 3.10 – Graphe de la courbe elliptique $E : y^2 = x^3 + 3x + 8$ sur $\mathbb{R}$

Si nous voulons tracer le graphe de la même courbe sur le corps $\mathbb{F}_{13}$, il apparait sous forme de points représentant l'ensemble des couples $(x, y) \in \mathbb{F}_{13}^2$ qui résolvent l'équation

$$E : y^2 = x^3 + 3x + 8 \mod 13$$

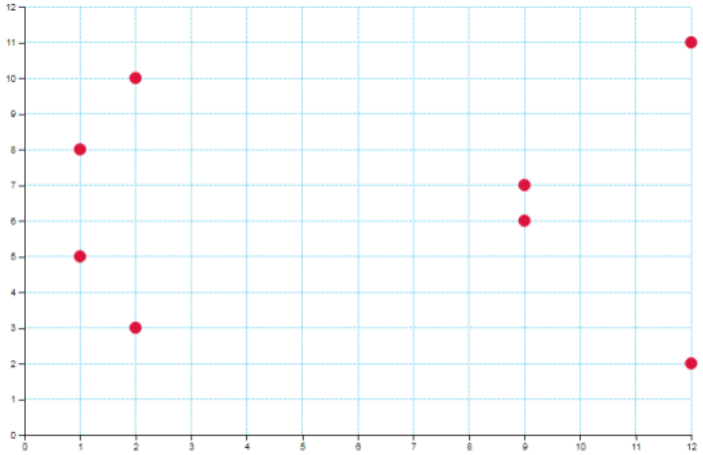

FIGURE 3.11 – Les points représentants la courbe $E : y^2 = x^3 + 3x + 8$ sur $mathbbF_{13}$

L'ensemble des points résolvant l'équation :

$$E(\mathbb{F}_{13}) = \{O, (1, 5), (1, 8), (2, 3), (2, 10), (9, 6), (9, 7), (12, 2), (12, 11)\}$$

La définition de toutes les combinaisons d'additions de points possibles sur ce corps, peut être présentée sous forme d'une table.

+	$\mathcal{O}$	(1,5)	(1,8)	(2,3)	(2,10)	(9,6)	(9,7)	(12,2)	(12,11)
$\mathcal{O}$	$\mathcal{O}$	(1,5)	(1,8)	(2,3)	(2,10)	(9,6)	(9,7)	(12,2)	(12,11)
(1,5)	(1,5)	$\mathcal{O}$	(2,10)	(9,6)	(9,7)	(12,2)	(12,11)	(2,3)	(1,8)
(1,8)	(1,8)	(2,10)	$\mathcal{O}$	(9,7)	(9,6)	(12,11)	(12,2)	(1,5)	(2,3)
(2,3)	(2,3)	(9,6)	(9,7)	$\mathcal{O}$	(1,8)	(2,10)	(12,11)	(1,5)	(12,2)
(2,10)	(2,10)	(9,7)	(9,6)	(1,8)	$\mathcal{O}$	(12,2)	(2,3)	(12,11)	(1,5)
(9,6)	(9,6)	(12,2)	(12,11)	(2,10)	(12,2)	$\mathcal{O}$	(1,5)	(1,8)	(9,7)
(9,7)	(9,7)	(12,11)	(12,2)	(1,8)	(2,3)	(1,5)	$\mathcal{O}$	(2,10)	(9,6)
(12,2)	(12,2)	(2,3)	(1,5)	(1,5)	(12,11)	(1,8)	(2,10)	$\mathcal{O}$	(9,6)
(12,11)	(12,11)	(1,8)	(2,3)	(12,2)	(1,5)	(9,7)	(9,6)	(9,6)	$\mathcal{O}$

TABLE 3.1 – Table d'addition des points pour $E : y^2 = x^3 + 3x + 8$ sur $\mathbb{F}_{13}$

Puisque une courbe elliptique définie sur un corps fini est représentée sous forme de points,est il possible de calculer le nombre de ces points qui résolvent l'équation ?

La réponse est oui, on l'appelle l'ordre d'une courbe elliptique. Cependant il est considérécomme un problème difficile de l'ordre O(p) si p est grand. Par contre, différents algorithmes ont vu le jour, parmi les plus utilisés et prenant un temps polynomial de calcul, celui de René Schoof qui se base sur un théorème limitant le cardinal de E dans un intervalle. Proposé par Helmut Hasse (1936), ce théorème donne un majorant et un minorant de l'ordre du groupe de points de la courbe [5] elliptique sur un corps fini :

Théorème (Hasse) Étant donné une courbe elliptique E sur $\mathbb{F}_p$, le nombre de points sur la courbe $\#E$ est délimité comme suit [7]

$$\#E(\mathbb{F}_p) = p + 1 - t_p \quad \text{avec} \quad |t_p| \leq 2\sqrt{p}$$

ce qui donne

$$p + 1 - 2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}$$

donc

$$t_p = p + 1 - \#E(\mathbb{F}_p)$$

Exemple 2 : Théorème de Hasse

Prenons la courbe de l'exemple précédent $E(\mathbb{F}_{13}) : y^2 = x^3 + 3x + 8 \pmod{13}$ et appliquons le théorème de Hasse :

$$13 + 1 - 2\sqrt{13} \leq \#E(\mathbb{F}_{13}) \leq 13 + 1 + 2\sqrt{13}$$

$$2\sqrt{13} \approx 7,21 \quad \text{donc :}$$

$$14 - 7,21 \leq \#E(\mathbb{F}_{13}) \leq 14 + 7,21$$

$$6,79 \leq \#E(\mathbb{F}_{13}) \leq 21,21$$

Effectivement, suivant la liste de l'ensemble des points déjà donnée pour cette courbe, en incluant $\mathcal{O}$, on compte 9 points et $6,79 \leq 9 \leq 21,21$.

Le théorème de Hasse sert largement à identifier le corps approprié pour un nombre de points précis dans une courbe elliptique, et ce pour des fins de sécurité purement cryptographiques. Par exemple, si nous avons besoin d'une courbe avec 2^{160} éléments, nous devons utiliser un nombre premier de longueur d'environ 160 bits. [1]

3.7 Groupe cyclique dans E

L'ensemble des points qui résolvent l'opération de multiple d'un point de E sur un corps fini $\mathbb{F}_p$, représente un groupe cyclique pour une courbe elliptique, tel que :

Étant donné deux points P et Q choisis de E, et partagés publiquement, la résolution du problème de logarithme discret revient à déterminer la valeur du scalaire n satisfaisant l'équation [6] :

$$Q = \underbrace{P + P + P + \cdots + P}_{n \text{ additions}} = n \cdot P \tag{3.2}$$

Exemple 3 : Problème du logarithme discret dans E

Soit la courbe elliptique $E : y^2 \equiv x^3 + 2x + 2 \pmod{17}$. L'ordre de cette courbe $\#E = 19$, et les points résolvants $E(\mathbb{F}_{17})$ forment un groupe cyclique.

Soit le point $P(5,1) \in E$, calculons tous les multiples de P :

$$\begin{aligned}
2P &= (5,1) + (5,1) \pmod{17} = (6,3) & 11P &= (13,10)\\
3P &= 2P + P = (10,6) & 12P &= (0,11)\\
4P &= (3,1) & 13P &= (16,4)\\
5P &= (9,16) & 14P &= (9,1)\\
6P &= (16,13) & 15P &= (3,16)\\
7P &= (0,6) & 16P &= (10,11)\\
8P &= (13,7) & 17P &= (6,14)\\
9P &= (7,6) & 18P &= (5,16)\\
10P &= (7,11) & 19P &= \mathcal{O}
\end{aligned}$$

On a $19P = \mathcal{O}$, donc l'ordre de P est $19 = \#E$, ce qui fait de P un générateur de notre ensemble de points résolvants l'équation E car tous les éléments de E ont été engendrés par P.

L'algorithme permettant de calculer nP, n'est qu'une version améliorée de celui d'exponentiation rapide. Il suffit de remplacer les opérations d'exponentiation et de multiplication dans $\mathbb{Z}$ par le doublement d'un point et l'addition de deux points dans $E(\mathbb{F}_p)$ respectivement.

1. Un groupe $(G, *)$ est un groupe cyclique s'il existe un élément $a \in G$ tel que : $\forall x \in G, \exists k \in \mathbb{Z}$ tel que $x = a^k$

L'algorithme prend en entrée le point P qu'on voudrait multiplier [5] ainsi que l'entier n qu'on convertit en binaire tel que :

$$n = n_0 \cdot 2^0 + n_1 \cdot 2^1 + n_2 \cdot 2^2 + \cdots + n_t \cdot 2^t \quad \text{avec} \quad n_0, n_1, n_2, \cdots, n_t \in \{0, 1\}$$

L'algorithme utilise deux registres (qu'on notera D et A) pour accumuler les résultats du dédoublement et d'addition respectivement, ensuite boucle en parcourant le scalaire n depuis le poids le plus fort au plus faible et teste chacune de ses valeurs (0 ou 1). Le registre D est initialisé par la valeur du point P, tandis que A par le point à l'infini car il représente l'élément neutre de l'opération d'addition dans E.

Algorithm 2 Multiplication d'un scalaire par un point

Entrées : P, n
Initialisation :
  $D \leftarrow P$
  $A \leftarrow \mathcal{O}$ *// $\mathcal{O}$ est le point à l'infini*
Sorties : A
Algorithme :
  $d = \sum_{i=0}^{t} n_i \cdot 2^i$
for $i = t$ **à** 0 **faire do**
    $D \leftarrow D + D \pmod{n}$
    if $n_i == 1$ **then**
      $A \leftarrow A + D \pmod{n}$
    end if
end for
Retourner A

3.8 Probleme du Logarithme Discret

La totalité des protocoles cryptographiques basés sur les courbes elliptiques reposent sur la difficulté de résoudre le problème du logarithme discret, car les opérations d'addition et de multiplication que nous venons d'entamer sur l'ensemble $E(\mathbb{Z}_p)$, sont plus complexes que leur équivalentes dans $\mathbb{R}$.

La sécurité des cryptosystèmes basés sur les courbes elliptiques repose sur le manque d'un algorithme de calcul qui pourrait résoudre le problème du logarithme discret elliptique en un temps polynomial. Autrement dit, dans moins de $O(\sqrt{p})$ étapes. Malgré la nature très structurée du groupe $E(\mathbb{F}_p)$, les algorithmes les plus rapides connus pour résoudre le problème du logarithme discret elliptique ne sont pas mieux que l'algorithme générique qui fonctionne aussi bien pour résoudre le problème du logarithme discret dans n'importe quel groupe. Il existe des courbes et des nombres premiers pour lesquels le problème du logarithme discret dans $E(\mathbb{F}_p)$ est comparativement facile. Ces cas particuliers sont à éviter dans la conception des cryptosystèmes sécurisés.

Soient G un groupe et $g \in G$. Le problème du logarithme discret dans G en base g

est, pour $y \in G$ donné, de trouver un entier x tel que

$$g^x = y$$

($xg = y$ si G est noté additivement).

Dans le cas où $G = E$ est une courbe elliptique, le problème du logarithme discret en base $P \in E$ est de trouver, étant donné $Q \in E$, un entier x tel que

$$Q = xP$$

si un tel x existe.[7]

3.9 Protocole d'échange de clé pour les courbes elliptiques

En cryptographie, on appelle échange de clé toute opération permettant à deux parties communicantes sur un canal non sécurisé, d'établir une clé secrète commune. Ce type de protocole exige des fonctions mathématiques dites à sens uniques. Le premier protocole ayant vu le jour a été publié par Whitfield Diffie et Martin Hellman en 1976, d'où son nom Diffie-Hellman. Il fut également le premier schéma cryptographique asymétrique.

Basé sur le problème du logarithme discret et se servant de la propriété de commutativité de l'exponentiation, le schéma de Diffie-Hellman apporta une solution à une variété de protocoles de communication réseau, tel que SSH (Secure Shell), IPSec (Internet Protocol Security) et TLS (Transport Layer Security) [5].

$$k = (\alpha^x)^y \equiv (\alpha^y)^x \pmod{p}$$

Alice et Bob veulent avoir une clé en commun pour s'échanger des données en toute sécurité. Supposons que leur seul moyen de communication soit public. Un des moyens de sécuriser leurs données est qu'ils établissent une clé privée entre eux. La méthode de Diffie-Hellman permet justement de faire cela (en général on utilise cette méthode avec des groupes $\mathbb{F}_q^*$, mais nous présentons cette méthode adaptée pour les courbes elliptiques).

On prend un point de départ $G(x_1, y_1)$ dans $E_q(a, b)$ dont l'ordre n est élevé. L'ordre n d'un point sur une courbe elliptique est le plus petit entier positif tel que $nG = \mathcal{O}$.

L'échange d'une clé par ECC entre deux entités A et B se déroule comme suit :

- A choisit un n_A inférieur à n qui sera sa clé privée. A génère alors sa clé publique

$$P_A = n_A \cdot G.$$

- B choisit un n_B inférieur à n qui sera sa clé privée. B génère alors sa clé publique

$$P_B = n_B \cdot G.$$

- A génère la clé secrète

$$K = n_A \cdot P_B$$

et B génère la clé secrète

$$K = n_B \cdot P_A.$$

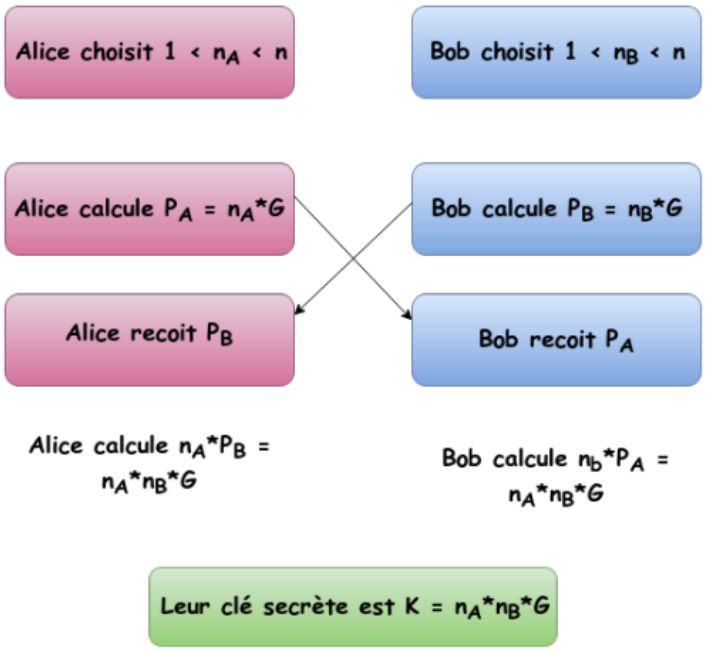

FIGURE 3.12 – Échange de clés Diffie-Hellman

Notons que K peut être utilisé comme clé secrète pour la cryptographie symétrique (AES, ...) [7].

3.10 Conclusion

Après avoir acquis les bases sur lesquelles repose un cryptosystème conçu par les courbes elliptiques, nous pouvons comprendre les avantages qu'offre un tel système. Malgré la complexité de ses opérations de base, ce système offre des performances élevées, ce qui est un point très fort vis-à-vis des environnements modernes à sécuriser tels que les réseaux, les bases de données, le cloud, les objets connectés, etc. Ces environnements présentent des contraintes variées, telles que la bande passante et l'énergie, exigeant moins de calculs et des clés moins volumineuses sans perte du niveau de sécurité.

CHAPITRE 4

ETAT DE L'ART

Sommaire

4.1 Introduction

Depuis les premières études sur les systèmes de cryptographie, le développement de méthodes de chiffrement sûres et efficaces n'a cessé de progresser. En particulier, le système de chiffrement à sac à dos (Knapsack Cryptosystem) a suscité un intérêt considérable, notamment lorsqu'il est combiné avec la cryptographie à courbes elliptiques (ECC). Depuis l'introduction des courbes elliptiques dans les protocoles de cryptographie, ces approches ont offert des avantages notables en termes de sécurité et de performance.

Dans cette état de l'art, nous présentons de multiples approches apportant chacune une solution différente à la technique de problème de sac à dos par les courbes elliptiques.

4.2 Koichiro Noro et Kunikatsu Kobayashi. 2009

Koichiro et Kunikatsu. proposent un cryptosystème Système de cryptographie de sac à dos utilisant les couplages de message basé sur les courbes elliptiques[9].

4.2.1 Couplage sur courbes elliptiques

Le couplage de Tate est utilisé, noté $e(P, Q)$ pour les points rationnels P, Q sur une courbe elliptique avec un degré d'insertion l.

4.2.2 Génération de clé

- Choisissez un nombre premier p (1024 bits ou plus)
- Définissez la courbe elliptique $E(F_p) : y^2 = x^3 + ax + b$.
- Assurez-vous que p est choisi de sorte que l'ordre de $E(F_p)$ inclut un grand nombre premier n (160 bits ou plus).
- Sélectionnez un point rationnel $P \in E(F_p)[n]$ et $Q \in E(F_{p^l})$.
- Calculez le couplage $e(P, Q)$. Choisissez une constante aléatoire k satisfaisant $\sum_{i=1}^{ur}(k \cdot 2^{i-1}) < (n-1)/2$.
- Rendez public a_iP pour $i = 1, 2, ..., ur$. La clé publique comprend $a_1P, ..., a_{ur}P, E(F_p), R, S, r$. La clé secrète comprend $d, f_i(x), e(P, Q), a_1, ..., a_{ur}$.

4.2.3 Chiffrement

- Représentez le texte clair comme un vecteur binaire $M = (m_1, m_2, ..., m_{ur})$.
- Calculez le texte chiffré $C = \sum_{i=1}^{ur} m_i(a_iP)$.
- Groupez les sommes $C_1, ..., C_{u-1}$ et chiffrez-les avec le chiffrement ElGamal sur les courbes elliptiques.

4.2.4 Déchiffrement

Récupérez C_i à partir de C_{i1}, C_{i2} en utilisant la clé secrète d. Calculez $e(C, Q)$ et $e(C_i, Q)$ pour décrypter le vecteur binaire M.

Validité du déchiffrement La méthode de déchiffrement est validée en vérifiant les valeurs de couplage et en s'assurant qu'elles correspondent aux bits de texte clair appropriés.

Complexité computationnelle Les opérations de chiffrement et de déchiffrement sont en temps polynomial grâce à l'efficacité du calcul des opérations sur les courbes elliptiques et des couplages.

Considérations de sécurité Le chiffrement ElGamal sécurise les étapes intermédiaires. La haute dimensionnalité du vecteur de sac à dos rend les attaques par force brute impraticables.

Exemple numérique Considérez la courbe elliptique $E(F_p) : y^2 = x^3 - x$ avec un nombre premier p de 163 bits. L'ordre de $E(F_p)$ inclut un nombre premier n de 143 bits. Les étapes de génération de clé, de chiffrement et de déchiffrement sont démontrées avec des valeurs explicites pour illustration.

4.3 R. Rajaram Ramasamy et M. Suguna et al. 2009

Rajaram Ramasamy et al proposent une approche innovante pour améliorer la sécurité de l'encryption en utilisant la Cryptographie à Courbes Elliptiques (ECC) avec l'algorithme de Knapsack. L'objectif est de développer un système de chiffrement plus sécurisé tout en maintenant une taille de clé réduite[11].

4.3.1 Méthodologie

— Utilise des courbes elliptiques définies sur des champs finis.

— L'opération principale est la multiplication scalaire d'un point par un entier.

4.3.2 Génération des Clés

Clé Privée (PR)

Une clé privée est générée de manière aléatoire dans un intervalle spécifié.

— Formule : $PR = \text{randint}(min\_val, max\_val)$

— Où "randint" est une fonction qui génère un nombre entier aléatoire entre "min_val" et "max_val".

Clé Publique (PB)

La clé publique est calculée en multipliant la clé privée par le générateur de la courbe elliptique.

— Formule : $PB = PR \times G$

— Où "G" est le générateur de la courbe elliptique.

4.3.3 Chiffrement

Conversion du Message en Points sur la Courbe Elliptique Le message est converti en une séquence de valeurs ASCII.

— Chaque valeur ASCII est transformée en un point sur la courbe elliptique.

— Formule : Pour chaque caractère C dans le message, le point correspondant est $P = C \times G$.

Chiffrement avec l'Algorithme de Knapsack Chaque point P est chiffré individuellement en utilisant l'algorithme de Knapsack avec la clé publique.

— Formule : Pour chaque point chiffré C', $C' = (P \times PB) + E$

— Où "E" est une erreur aléatoire introduite pour renforcer la sécurité.

4.3.4 Déchiffrement

Déchiffrement avec la Clé Privée Chaque point chiffré est déchiffré en utilisant la clé privée.

— Formule : Pour chaque point chiffré C', $P = \frac{(C'-E)}{PR}$

— Où "E" est soustraite pour annuler l'erreur introduite lors du chiffrement.

Conversion des Points Déchiffrés en Message Original Les points déchiffrés sont convertis en valeurs ASCII.

— Ces valeurs ASCII sont alors reconstituées pour former le message d'origine.

4.4 Synthèse

Les approches étudiées ont toutes réussi à développer un système de problème de sac à dos en utilisant les courbes elliptiques. Bien que chacune de ces méthodes présente des avantages en ajoutant des options au schéma initial, quelques inconvénients ont été également remarqués.

L'approche proposée par Koichiro Noro et Kunikatsu Kobayashi. 2009 présente un système de cryptographie hybride qui intègre les courbes elliptiques et le problème du sac à dos. Cette méthode exploite les avantages de la sécurité des courbes elliptiques et de la complexité du problème du sac à dos pour renforcer la cryptographie. Parmi les avantages, ce système offre une sécurité élevée avec des clés plus courtes, ce qui améliore l'efficacité et réduit les besoins en stockage par rapport à des systèmes traditionnels comme RSA. De plus, la complexité accrue rend les attaques plus difficiles. Cependant, l'implémentation de ce système peut être plus complexe et nécessite une compréhension approfondie des deux domaines cryptographiques. Les recherches futures sont nécessaires pour optimiser cette approche et évaluer sa robustesse face aux différentes menaces.

L'approche proposée par R. Rajaram Ramasamy et M. Suguna et al. 2009 propose une méthode de sécurisation des données combinant la cryptographie à courbes elliptiques (ECC) et l'algorithme du sac à dos (knapsack). Cette méthode convertit les valeurs ASCII des caractères en points affines sur une courbe elliptique, puis applique l'algorithme du sac à dos pour accroître la sécurité. Parmi les avantages, cette approche nécessite la confidentialité de moins de paramètres que des méthodes comme RSA et rend le déchiffrement

pratiquement impossible même si certains paramètres sont compromis. Cependant, elle exige plus de temps et d'espace pour l'implémentation, ce qui peut être un inconvénient dans certaines applications. L'article conclut en suggérant des pistes pour améliorer cette méthode et explore ses applications potentielles dans la cryptographie moderne.

4.5 Conclusion

Dans cet état de l'art, nous avons présenté différents types d'approches comportant une variété de problème de sac dos dans le domaine de partage de secret en utilisant les courbes elliptiques. C'est grâce à cette étude qu'on a été inspiré à établir notre propre approche pour développer une application de simulation adéquate.

CHAPITRE 5

CONTRIBUTION

Sommaire

5.1 Introduction

Dans le chapitre précédent, nous avons exposé quelques approches concernant deux types différents de problèmes de sac à dos utilisant les courbes elliptiques. À présent, nous allons présenter une méthode pour renforcer le chiffrement Merkle-Hellman en y intégrant des courbes elliptiques, augmentant ainsi la sécurité contre les attaques. Ce procédé se déroule en trois étapes, comme tout chiffrement. Nous décrirons également les démarches d'implémentation suivies ainsi que les résultats obtenus.

5.2 Les outils Utilisés

5.2.1 Définition Java

Le langage de programmation Java est très populaire, et largement utilisé pour la création d'applications desktop, web ou mobiles hautement performantes et sécurisées. Les IDE Java offrent aux développeurs une large gamme d'outils de développement logiciel. Ils sont conçus pour fonctionner avec des plateformes d'applications spécifiques et réduire le cycle de développement logiciel. Les meilleurs IDE Java sont IntelliJ IDEA, Visual Studio, Eclipse, Xcode et NetBeans. et en a utilisé NetBeans

5.3 Codage

Le codage de message entré par l'utilisateur utilise un dictionnaire que nous avons créé nous-mêmes. Ce dictionnaire contient les alphabets de 'a' à 'z', en notant que les lettres majuscules et minuscules prennent la même valeur. Il inclut également l'espace et les chiffres de 0 à 9. Pour coder le message, nous prenons chaque caractère un par un, trouvons la valeur correspondante dans le dictionnaire, et remplaçons chaque caractère par sa valeur correspondante. Après cette étape, nous obtenons une chaîne de nombres.

Pour ne pas perdre l'information, nous mettons ces nombres dans un tableau. Ensuite, nous convertissons chaque nombre en binaire afin de pouvoir chiffrer le message. Cependant, pour éviter des ambiguïtés lors de la conversion en binaire, il faut fixer la taille de chaque nombre binaire. si le message est "c1", il devient 12,1. En binaire, 12 est représenté par 1100 et 1 par 1. Si nous concaténons ces deux valeurs en une seule chaîne, nous obtenons 11001. Cela peut poser problème lors de la reconstitution du message initial car 11001 pourrait être interprété comme 1 (qui est 1 en décimal) et 1001 (qui est 9 en décimal), donnant ainsi 1,9 au lieu de 12,1. Pour résoudre ce problème, nous devons garder une taille fixe pour chaque nombre binaire. Par exemple, avec une taille fixe de 4 bits, le message devient 1100 pour 12 et 0001 pour 1. Ainsi, le message binaire final sera 11000001.

Interprété comme 1 (qui est 1 en décimal) et 1001 (qui est 9 en décimal), donnant ainsi 1,9 au lieu de 12,1. Pour résoudre ce problème, nous devons garder une taille fixe pour chaque nombre binaire. Par exemple, avec une taille fixe de 4 bits, le message devient 1100 pour 12 et 0001 pour 1. Ainsi, le message binaire final sera 11000001.

L'alphabet (Partie 1)		L'alphabet (Partie 2)	
Codage	Alphabet	Codage	Alphabet
0	A_a	19	T_t
1	B_b	20	U_u
2	C_c	21	V_v
3	D_d	22	W_w
4	E_e	23	X_x
5	F_f	24	Y_y
6	G_g	25	Z_z
7	H_h	26	Espace
8	I_i	27	!
9	J_j	28	.
10	K_k	29	,
11	L_l	30	?
12	M_m	31	39
13	N_n	32	40
14	O_o	33	
15	P_p	34	
16	Q_q	35	
17	R_r	36	
18	S_s	37	

TABLE 5.1 – Tableau de Dictionnaire

5.3.1 Création du Dictionnaire

Le dictionnaire que nous avons utilisé est :

5.4 Utilisation des courbes elliptiques pour le chiffrement

Pour renforcer ce chiffrement, nous utilisons les courbes elliptiques en ajoutant des paramètres spécifiques.

5.4.1 Définition des paramètres de la courbe elliptique

Sélectionnez les paramètres a, b et le modulo p pour définir la courbe elliptique $y^2 = x^3 + ax + b \mod p$. Par exemple, choisissons $a = 2$, $b = 3$, et $p = 17$. L'équation de la courbe devient :

$$y^2 = x^3 + 2x + 3 \mod 17$$

5.5 Chiffrement de Merkle-Hellman basé sur les courbes elliptiques

Dans le chiffrement de Merkle-Hellman, après avoir converti le message en binaire et calculé la somme en multipliant chaque bit par le b_i correspondant, le résultat de cette somme est transmis sous forme de point appartenant à la courbe elliptique. Cette méthode assure que le message chiffré peut être représenté et manipulé efficacement en utilisant les propriétés des courbes elliptiques pour une sécurité renforcée. en suivre les étapes de cet algorithme

5.5.1 Étape 1 : Génération de clés

Génération de la suite supercroissante Une suite supercroissante est une séquence de nombres où chaque élément est strictement supérieur à la somme de tous les éléments précédents. Par exemple, si $a_1 = 2$, $a_2 = 3$ (car $3 > 2$), $a_3 = 7$ (car $7 > 2 + 3$), etc.

Calcul de N Choisissez un nombre N tel que N soit supérieur à la somme de tous les éléments de la suite. Calculer N tel que $N > \sum_{i=1}^{n} a_i$. Par exemple, si notre suite est $2, 3, 7, 14$, alors N doit être supérieur à $2 + 3 + 7 + 14 = 26$. Supposons $N = 31$.

Choix de A Choisissez un nombre A tel que $\gcd(A, N) = 1$. Par exemple, si $N = 31$, nous pouvons choisir $A = 3$.

Calcul des b_i Pour chaque i de 1 à n, calculer chaque b_i comme $b_i = a_i \times A \mod N$. par exemple

i	a_i	$b_i = a_i \times 3 \mod 31$
1	2	$2 \times 3 \mod 31 = 6$
2	3	$3 \times 3 \mod 31 = 9$
3	4	$4 \times 3 \mod 31 = 12$
4	5	$5 \times 3 \mod 31 = 15$
5	6	$6 \times 3 \mod 31 = 18$
6	7	$7 \times 3 \mod 31 = 21$
7	8	$8 \times 3 \mod 31 = 24$
8	9	$9 \times 3 \mod 31 = 27$
9	10	$10 \times 3 \mod 31 = 30$
10	11	$11 \times 3 \mod 31 = 2$

TABLE 5.2 – Calculs de b_i pour différentes valeurs de a_i

Génération de k Choisissez un nombre aléatoire k, par exemple $k = 7$. Dans le cadre des nombres ordinaires, générer un nombre aléatoire k et calculer son inverse est très facile. Nous générons un nombre k aléatoire et calculons son inverse en utilisant l'algorithme d'Euclide étendu ou par recherche exhaustive (en multipliant ce nombre par des nombres inférieurs au modulo jusqu'à obtenir 1).

Par exemple, 2 est l'inverse de 9 dans le modulo 17 car $2 \times 9 = 18 \equiv 1 \mod 17$. Si nous choisissons un nombre quelconque, disons 3, et le multiplions par 2, $3 \times 2 = 6 \mod 17$. Ensuite, $6 \times 9 = 54 \equiv 3 \mod 17$. Nous retrouvons ainsi le même nombre.

Cependant, dans le cas des points sur une courbe elliptique, cette relation n'existe pas. Si nous multiplions un point P par k, nous obtenons un autre point P_1. Pour retrouver le point original à partir de P_1, il n'est pas possible de revenir directement au point de départ. En considérons l'équation $y^2 = x^3+2x+3 \mod 17$ et le point $(2, 7)$ qui appartient à cette courbe. Si nous multiplions ce point par 2, nous obtenons $(2, 7) \times 2 = (14, 15)$. Cependant, si nous multiplions le point $(14, 15)$ par 9, nous obtenons $(14, 15) \times 9 = (15, 12)$, qui n'est pas égal à $(2, 7)$. Cela illustre que la multiplication d'un point par un nombre sur une courbe elliptique ne permet pas de revenir facilement au point d'origine.

Pour cette raison, nous utilisons une méthode différente pour calculer l'inverse d'un nombre sur les courbes elliptiques. Cette méthode consiste à choisir un point aléatoire P_1 sur la courbe elliptique et à le multiplier par k, ce qui donne un nouveau point P_2 tel que $P_1 \times k = P_2$. Nous conservons P_1 et, pour trouver l'inverse de k, nous ajoutons P_2 de manière itérative jusqu'à retrouver le point P_1. À chaque itération, nous augmentons une variable invk de 1 jusqu'à ce que invk $\times P_2 = P_1$.

Après ces étapes, la clé publique est représentée par $(b_i, \text{invk}, a, b, z)$, où a, b et z sont échangés secrètement via le protocole de Diffie-Hellman. La clé privée dérivée est représentée par (a_i, A, N, k).

5.5.2 Étape 2 : Chiffrement

Le chiffrement s'effectue selon la méthode suivante :

1. Codé ou Convertir le message en binaire.
2. Pour chaque bit du message binaire, multiplier par le b correspondant et calculer la somme. Par exemple, pour le message "0101", cela donnerait :

$$0 \times b_0 + 1 \times b_1 + 0 \times b_2 + 1 \times b_3$$

3. Transmettre le résultat de la somme sous forme de point sur la courbe elliptique. et multiplier le point par invk pour obtenir le résultat final à transmettre.

5.5.3 Étape 3 : Déchiffrement

Le point reçu après le chiffrement est multiplié par k pour retrouver le point initial. et Utiliser le composant x du point résultant pour effectuer le déchiffrement de Merkle-Hellman.

5.6 Implémentation

Dans cette section nous présentons notre implémentation de l'approche proposée.

on a le message m = 12 alors en prend chaque caractere et en voir leur valeur dans le dictionnaire et en mettre cette valeur dans un tableau, mais dans ce cas en juste des nombres et dans notre dictionnaire les nombres reste les mêmes alors c'est pour ca notre tableau contient la même valeur de m=12 donc :

Algorithm 3 Algorithme de chiffrement de Merkle-Hellman

Initialisation des Variables :
$message \leftarrow$ demander à l'utilisateur de saisir un message
$car1 \leftarrow$ tableau de chaînes de caractères de taille longueur($message$)
$car \leftarrow$ tableau d'entiers de taille longueur($message$)
Conversion du Message :
for i de 0 à longueur($message$) $-$ 1 **do**
    $c \leftarrow$ message.charAt(i)
    $car[i] \leftarrow$ dictionnaire.get(c)
    afficher "car" + i + " = " + $car[i]$
    $car1[i] \leftarrow$ conv_binaire($car[i]$)
end for
Retourner le Résultat :
retourner $car1$

1	**2**

Apres en convertir ces nombres en binaire ,mais il faut que la taille de chaque nombre en binaire égale à 6 car on a le nombre maximale de dictionnaire = 40 et si en convertir ce nombre en binaire = 101000 donc la taille de grand nombre de dictionnaire = 6 c'est pour ca en fixe la taille des nombres en binaire de 6

1	**2**
000001	**000010**

Apres ça en convertir ce tableau en String qui égale à '000001000010'

5.6.1 L'algorithme de génération de clé

Avant de participer sur l'algorithme il faut de parler un petit peut sur la génération des nombres de la suite comment elle est organisée, je veux parler superficielle sur ces notions car on ait parler clairement sur ça dans le chapitre de chiffrement de merkle-hellman. Soient A_i les variables de la suite supercroissante. On a donc : Pour les séries A_i, nous avons les inégalités suivantes :

$$A_2 > A_1$$

$$A_3 > A_1 + A_2$$

$$A_4 > A_1 + A_2 + A_3$$

$$\vdots$$

$$A_n > A_1 + A_2 + \ldots + A_{n-1}$$

Si on reformule la syntaxe dans une autre formule, on trouve :

$$A_{n-1} = A_1 + A_2 + A_3 + \ldots + A_{n-2} + m$$

où m est un nombre aléatoire. Donc :

$$A_n = A_1 + A_2 + A_3 + \ldots + A_{n-1} + m_1$$

En reformulant, on obtient :

$$A_n = A_1 + A_2 + A_3 + \ldots + A_{n-2} + A_1 + A_2 + A_3 + \ldots + A_{n-2} + m + m_1$$

D'où :

$$A_n = 2 \cdot (A_1 + A_2 + A_3 + \ldots + A_{n-2}) + m + m_1$$

Sachant que :

$$A_1 + A_2 + A_3 + \ldots + A_{n-2} = A_{n-1} - m$$

on a :

$$A_n = 2 \cdot (A_{n-1} - m) + m + m_1$$

et donc :

$$A_n = 2 \cdot A_{n-1} - 2m + m + m_1$$

finalement :

$$A_n = 2 \cdot A_{n-1} - m + m_1$$

Exemple Si l'équation de la courbe $y^2 = x^3 + 2x + 2 \pmod{17}$ donc

$$\text{coord}[0] = 2, \quad \text{coord}[1] = 2, \quad \text{coord}[2] = 17$$

Alors en génère la suite super-croissante : En choisit $n = 6$.

$T[0]$	$T[1]$	$T[2]$	$T[3]$	$T[4]$	$T[5]$	$T[6] = N$
2	3	6	13	28	53	111

TABLE 5.3 – génération de la suite super-croissante

En choisit A = 2, avec la valeur K = 3 et il faut calculer et calculer l'inverse il faut choisir un point qui est inclut dans la courbe. Les points sont :

(0,6)	(0,11)	(3,1)	(3,16)	(5,1)	(5,16)	(6,3)	(6,14)	(7,6)
(7,11)	(9,1)	(9,16)	(10,6)	(10,11)	(13,7)	(13,10)	(16,4)	(16,13)

TABLE 5.4 – Les points de courbe pour calculer l'inverse de K

Et nous choisissons le point $(0, 6)$.

$$\text{Point } 1 = (0, 6) \times 3 = (6, 3)$$

Algorithm 4 Algorithme de génération de clés

Initialisation des paramètres :
$T \leftarrow$ tableau de la suite super-croissante
$n \leftarrow$ la taille de la suite super-croissante
$N \leftarrow$ nombre supérieur à la somme de tous les nombres de la suite super-croissante
$A \leftarrow$ nombre premier avec N, tel que $\text{PGCD}(A, N) = 1$
$K \leftarrow$ nombre entier généré aléatoirement
Coord $\leftarrow$ tableau des paramètres de l'équation de la courbe
$B \leftarrow$ tableau d'entiers
$\text{Inv}_K \leftarrow$ l'inverse de K
La taille du tableau T est égale à $n + 1$ car on ajoute le nombre N dans la dernière case du tableau
Génération de la suite super-croissante :
$T[1] \leftarrow$ nombre aléatoire
$T[2] \leftarrow T[1] +$ un autre nombre aléatoire ▷ m et $m1$ sont des valeurs générées aléatoirement
for $i \leftarrow 2$ to $n + 1$ **do**
    $T[i] \leftarrow 2 \cdot T[i - 1] - m + m1$
end for
$N \leftarrow 2 \cdot T[n] + m2$
$T[n + 1] \leftarrow N$
$A \leftarrow$ nombre_premier(N)
$K \leftarrow$ random()
Point $\leftarrow$ choisir_un_point_dans_la_courbe(Coord)
Point1 $\leftarrow$ multiplication_point(Point, K)
$\text{Inv}_K \leftarrow 0$
while Point2 $\neq$ Point1 **do**
    Point2 $\leftarrow$ addition_point(Point1)
    $\text{Inv}_K \leftarrow \text{Inv}_K + 1$
end while
for $i \leftarrow 1$ to n **do**
    $B[i] \leftarrow T[i] \cdot A \cdot T[n + 1]$
end for
Clé_pub(B, Inv_K, Coord) ▷ Coord envoyé secrètement
Clé_priv(T, A, K, Coord)

B[0]	B[1]	B[2]	B[3]	B[4]	B[5]
4	6	12	26	56	106

TABLE 5.5 – tableau de calcule de Bi

Aprés nous avons trouvé l'inverse de K :

$$\begin{aligned}
&(6,3)+(6,3)=(3,1) && \text{Inv_}K=2\\
&(3,1)+(6,3)=(16,13) && \text{Inv_}K=3\\
&(16,13)+(6,3)=(13,7) && \text{Inv_}K=4\\
&(13,7)+(6,3)=(7,11) && \text{Inv_}K=5\\
&(7,11)+(6,3)=(0,11) && \text{Inv_}K=6\\
&(0,11)+(6,3)=(9,1) && \text{Inv_}K=7\\
&(9,1)+(6,3)=(10,11) && \text{Inv_}K=8\\
&(10,11)+(6,3)=(5,16) && \text{Inv_}K=9\\
&(5,16)+(6,3)=(5,1) && \text{Inv_}K=10\\
&(5,1)+(6,3)=(10,6) && \text{Inv_}K=11\\
&(10,6)+(6,3)=(9,16) && \text{Inv_}K=12\\
&(9,16)+(6,3)=(0,6) && \text{Inv_}K=13
\end{aligned}$$

Donc Inv_$K=13$. Maintenant nous calculons les B_i :

$$\begin{aligned}
&B[0]=T[0]\times A[N]=2\times 2=4 \quad [111]\\
&B[1]=T[1]\times A[N]=3\times 2=6 \quad [111]\\
&B[2]=T[2]\times A[N]=6\times 2=12 \quad [111]\\
&B[3]=T[3]\times A[N]=13\times 2=26 \quad [111]\\
&B[4]=T[4]\times A[N]=28\times 2=56 \quad [111]\\
&B[5]=T[5]\times A[N]=53\times 2=106 \quad [111]
\end{aligned}$$

Le tableau de B est le suivant :

Donc la clé publique est : $(B, \text{Inv\_}K, \text{coord})$. La clé privée est : (T, A, K, coord). On peut désormais chiffrer le message à l'aide de la clé publique de la manière suivante :

5.6.2 Algorithme de Chiffrement

Algorithm 5 Algorithme de chiffrement

Déclaration des variables
$Message1$: tableau des nombres entiers binaires

function CHIFFREMENT(message)
    $Message1 \leftarrow$ Convertir_message_tableauentier($message$)
    $Somme \leftarrow 0$
    $J \leftarrow 0$
    $U \leftarrow 0$
    $Liste \leftarrow$ liste vide
    for $i \leftarrow 0$ **to** taille_message **do**
        $Somme \leftarrow Somme + Message1[i] \times B[J]$
        if $i \geq$ longueur(B) **and** taille_message $>$ longueur(B) **then**
            $J \leftarrow 0$
        end if
        $ind \leftarrow \frac{Somme}{\text{coord}[2]}$
        $Somme \leftarrow Somme\%\text{coord}[2]$
        while $Y == -1$ **do**
            $Y \leftarrow$ calc_y($Somme$, coord)
            $Somme \leftarrow Somme + 1$
            $U \leftarrow U + 1$
        end while
        $Point \leftarrow (Somme, Y)$
        $Point\_Chiffr \leftarrow$ multiplication_point_scalaire($Point$, Inv_K)
        Ajouter ce point à la liste avec U et ind
    end for
    On envoie ensuite la liste au destinataire
end function

En continuant avec l'exemple précédent :

— Message = 000001000010

— Message1 = 000001000010

Étant donné que le message a une taille de 12 bits et que la suite est super-croissante, nous avons divisé le message en deux parties : 000001 et 000010, puis nous calculons chaque partie séparément.

La 1ère partie : 000001 Somme = 0*4 + 0*6 + 0*12 + 0*26 + 1*56 + 0*106 = 56
Somme = 56 [17] = 5 [17]
Ind = somme / 17 = 3
Le Point est (5, 1) avec U = 0
Point1 = (5,1) * 13 = (16,4)
Nous avons ajouté le point (16,4) et le nombre 0 et 3 dans la liste.

La 2ème partie : 000010 Somme = 0*4 + 0*6 + 0*12 + 0*26 + 1*56 + 0*106 = 56
Somme = 56 [17] = 5 [17]
Ind = somme / 17 = 3
Le Point est (5, 1) avec U = 0

Point1 = (5,1) * 13 = (16,4)
Nous avons ajouté le point (16,4) et le nombre 0 et 3 dans la liste.

1ere partie	(16,4)	1	6
2eme partie	(16,4)	0	3

5.6.3 Algorithme de Déchiffrement

Algorithm 6 Algorithme de déchiffrement

Déclaration des variables : P : tableau d'entier $Tableau$: tableau d'entier qui contient les points originaux $Message\_chiff$: tableau d'entier qui contient la valeur de la Somme **L'algorithme :** $i = 1$ $taille\_liste$ $Tableau[i] = multiplication\_point(liste(i), k)$ $Message\_chif[j] = t3[0] - liste(i+3)$ $Message\_chif[j] = coord[2] * liste(i+2) + Message\_chif[j]$ $j \leftarrow j+1$ $i \leftarrow i+2$ // Le déchiffrement de Merkle-Hellman $Inv\_A = Calcul\_Inverse(A)$ $i = 1$ $taille(Message\_chif)$ $P[i] = Inv\_K * Message\_chif[i]\%T[n+1]$ $Moins = P[i]$ $Moins \neq 0$ $j = n-1$ 0 $Moins > T[j]$ $Moins \leftarrow Moins - T[j]$ Ajouter cette valeur dans une liste $Message1$ // Transmettre ce message en String $Message = String(Message1)$ Décodage($Message$) Afficher le message **Fin**

Exemple L'autre côté recevez la liste1 qui contient :

1ere partie	(16,4)	1	6
2eme partie	(16,4)	0	3

Donc pour déchiffrer ce message il faut déchiffrer partie par partie et trouver d'abord le point original :

1ère partie : (16, 4) * K = (16, 4) * 3 = (5, 1) nous prions juste le x : 5 – 1 = 4 et après nous avons multiplié 17 par 6 et ajouté ce résultat pour voir la valeur de message originale : 6 * 17 + 4 = 106 Donc si on trouve ce numéro on applique le déchiffrement de Merkle-Hellman :

Calcule de l'inverse de A mod N : Pour calculer le message chiffré dans le chiffrement de Merkle-Hellman on fait la somme des B_i de la façon suivante : Message = 01101 B_i = [B_0, B_1, B_2, B_3, B_4] Donc le message chiffré = $B_1 + B_2 + B_4$ ..... (1) En détaillant cela : $B_i = A * T[i] \mod 111$ Alors en remplaçant cette expression dans (1) on trouve : Message chiffré = $A*T[1] + A*T[2] + A*T[4]$ Message chiffré = $A*(T[1] + T[2] + T[4])$ et pour avoir juste les T pour faire la soustraction, il faut multiplier cette quantité par l'inverse de A :

Inv_A * Message chiffré = $\text{Inv}_A * A * (T[1] + T[2] + T[4])$ Inv_A * Message chiffré = $T[1] + T[2] + T[4]$ En entamant sur le calcul $A = 2, N = 111$ $2 * 56 = 112 \equiv 1 \mod 111$ d'ou 56 est l'inverse de 2 dans le modulo 111. après avoir trouvé l'inverse de A, on multiplie ce résultat par 106 : $\text{Inv}_A * 106 = 56 * 106 = 5936 \equiv 53 \mod 111$

Ensuite on fait la soustraction de la plus grande valeur jusqu'à la plus petite valeur dans T : $53 - 53 = 0$ Le message de la 1ère partie est : 000001.

2ème partie : En fait les mêmes étapes de la 1ère partie et donc on trouve : 000010. le message = 000001000010 après on fait le décodage de ce message par 6 bits : 000001 = 1 et 000010 = 2. Donc le message = 12

5.7 Application

Nous allons présenté notre Aplication avec la fenêtre principale, l'utilisateur doit entrer le message à chiffrer ainsi que les coordonnées de l'équation, puis cliquer sur le bouton "Valider", comme illustré dans l'image.

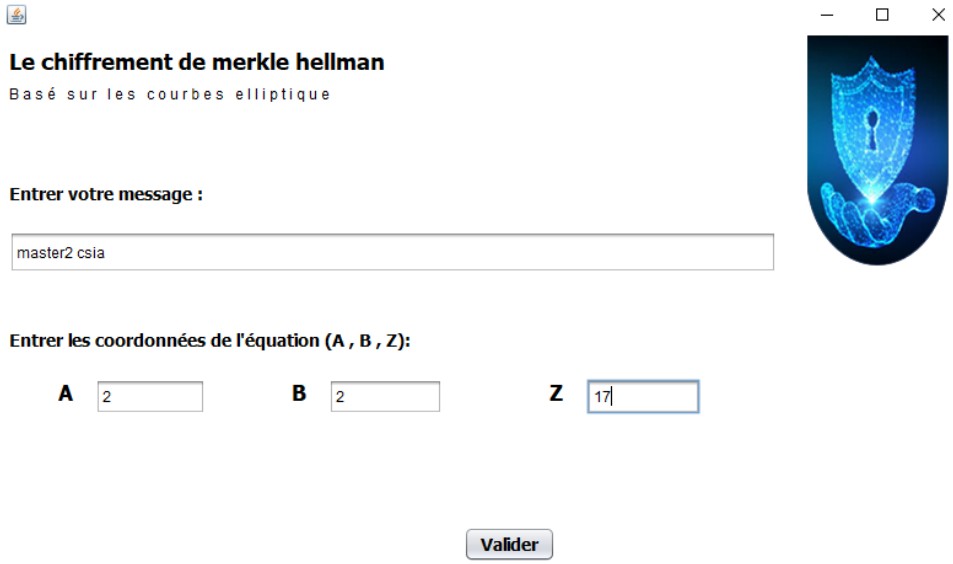

FIGURE 5.1 – l'interface de fenètre principale

Après avoir cliqué sur le bouton, trois fenêtres s'affichent :

Première fenêtre : affiche les clés utilisées (clé privée et clé publique) ainsi que les coordonnées de l'équation utilisée.

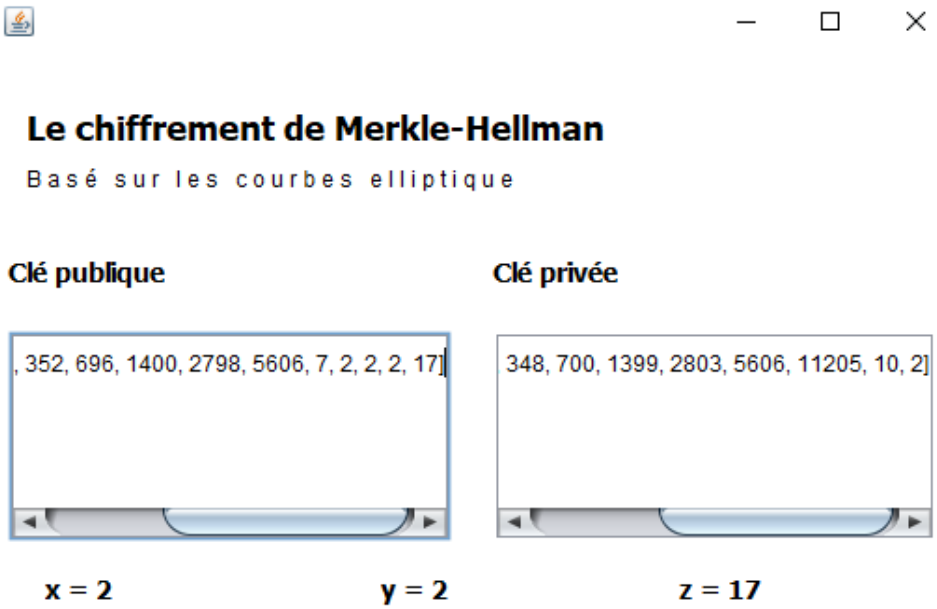

FIGURE 5.2 – Affichage des clefs

Deuxième fenêtre : affiche le message chiffré.

FIGURE 5.3 – Message chiffré

troixième fenetre : affiche le message déchiffré binaire et normale

FIGURE 5.4 – affichage de message chiffré et binaire

5.8 Synthèse des problèmes

Dans notre recherche sur ce sujet, nous avons rencontré plusieurs problèmes. Avant d'examiner cette solution, nous proposons d'autres solutions qui se présentent comme suit

5.8.1 Première solution

Génération des clés :

1. Générer une suite super-croissante de taille n telle que $a_i > a_{i-1}+a_{i-2}+\cdots+a_2+a_1$.
2. Générer un nombre N tel que N soit supérieur à la somme de toutes les valeurs de la suite super-croissante.
3. Choisir un nombre aléatoire A tel que $\text{PGCD}(A, N) = 1$.
4. Calculer les valeurs B_i telles que $B_i = A \times a_i \mod N$.
5. Choisir une équation d'une courbe elliptique quelconque et calculer les y correspondants pour chaque B_i de la suite. Par exemple : $y^2 = B_1^3 + B_1 \cdot a + b \cdot B_1 \mod z$.
6. Vérifier si ces valeurs existent sur la courbe. Si une valeur n'existe pas, ajouter 1 à a_i et recalculer B_i. Répéter ce processus jusqu'à trouver le bon point sur la courbe.
7. Si le point est trouvé, mettre à jour les valeurs de la suite super-croissante qui sont supérieures à a_i, recalculer les B_i correspondants, et répéter ces étapes jusqu'à compléter toute la suite et trouver tous les points correspondants aux B_i.

Ainsi, la clé publique est constituée des points (B_i, y_i) et la clé privée est composée de (a_i, N, A).

Le chiffrement :

— Pour chiffrer un message avec cette méthode, il faut additionner tous les points correspondant aux bits du message qui sont égaux à 1.

— Par exemple, si le message est 01101 et que la clé publique est $\{(B1, y1), (B2, y2), (B3, y3), (B4, y4)$ alors pour chiffrer ce message, on procède comme suit :

$$\text{message\_chiffré} = (B2, y2) + (B3, y3) + (B5, y5)$$

— Ensuite, on transmet le point message_chiffré.

Déchiffrement : Pour le déchiffrement, nous utilisons l'algorithme Baby-Step Giant-Step (BSGS) pour trouver les points qui composent la somme donnée. Une fois ces points trouvés, nous n'avons besoin que des B_i pour effectuer le déchiffrement de Merkle-Hellman.

Cependant, cette méthode présente une faiblesse majeure : un attaquant peut facilement déchiffrer le message en utilisant également l'algorithme BSGS, ce qui lui permettrait de voir directement le message. De plus, cette méthode ne fait pas pleinement usage de la clé privée. En effet, l'attaquant pourrait interrompre le processus de déchiffrement après avoir utilisé l'algorithme BSGS, sans avoir besoin de poursuivre avec le déchiffrement de Merkle-Hellman, puisqu'il pourrait déjà trouver le message. En conséquence, nous n'approuvons pas cette méthode, car elle présente des vulnérabilités similaires à celles de la méthode LLL et n'exploite pas entièrement la clé privée.

5.8.2 Deuxième Solution

1. **Génération des clés :**
   — Nous générons une suite super-croissante de taille n.
   — Ensuite, nous choisissons une équation de courbe elliptique et un point P appartenant à cette courbe. Et nous calculons les B_i de la manière suivante : $B_i = a_i \times P$.
   — Ainsi, la clé publique est constituée des B_i, et la clé privée est composée des a_i et du point P.
2. **Le chiffrement :** Pour le chiffrement, nous suivons un processus similaire à la méthode précédente en additionnant tous les points correspondant à un bit de message égal à 1. Par exemple, si le message est 01101, alors le message chiffré C est calculé comme la somme des points B_i correspondant aux bits à 1 dans le message, donc $C = B_2 + B_3 + B_5$. Ensuite, nous envoyons cette valeur C comme message chiffré.
3. **Déchiffrement** Pour déchiffrer le message dans cette méthode, nous sommes confrontés au problème du logarithme discret sur les courbes elliptiques. En effet, étant donné que le message chiffré C est la somme des points B_i correspondant à des bits de message égaux à 1, nous avons $C = \sum a_i \times P$, où a_i sont les coefficients et P est le point de la courbe elliptique. En reprenant l'exemple précédent, si $C = B_2 + B_3 + B_5$, alors $C = a_2 \times P + a_3 \times P + a_5 \times P = (a_2 + a_3 + a_5) \times P$. Nous avons donc C et P, et nous devons trouver $a_2 + a_3 + a_5$, ce qui revient à résoudre le problème du logarithme discret $C = K \times P$ pour $K = a_2 + a_3 + a_5$.

5.8.3 Troisième Solution

Dans notre méthode, nous avons utilisé une approche exhaustive pour la recherche de l'inverse_k. Nous avons cherché une méthode directe pour trouver l'inverse d'un nombre sur les courbes elliptiques, mais sans succès. C'est pourquoi nous avons proposé cette méthode. Nous avons gardé les paramètres de l'équation de la courbe elliptique (a, b, z) secrets et avons choisi des nombres plus petits que z (le modulo de l'équation) afin de ne pas avoir un temps d'exécution trop long lors de la recherche de l'inverse_k. Si nous trouvons une relation exacte permettant de calculer l'inverse_k, alors nous pourrons afficher les paramètres de l'équation de la courbe et choisir n'importe quelle valeur de modulo z. C'est le principal problème qui persiste dans nos recherches : trouver une relation exacte pour calculer l'inverse de k sur les courbes elliptiques. Ce défi demeure crucial pour nos travaux.

5.8.4 Les problèmes restants

est comme mentionné précédemment, nous avons utilisé une méthode exhaustive pour la recherche de l'inversek, mais cette méthode ne fonctionne pas pour toutes les équations des courbes elliptiques. En effet, au cours de nos recherches, nous avons observé qu'il existe 2 types de courbes elliptiques. Pour mieux comprendre, nous avons fourni un exemple de ces types.

type 1 Le premier type est le suivant : dans les courbes elliptiques où l'inverse de k est le même pour tous les points de la courbe, par exemple, si l'on prend un point $p1$ et que l'on calcule $p1 \times k = p2$, alors l'inverse, $p2 \times \text{inv\_}k = p1$, est valide pour tous les points de la courbe. Ainsi, si l'on prend n'importe quel point de cette courbe et que l'on le multiplie par k, pour retrouver le point original à partir du point résultant de la multiplication, il suffit de multiplier ce dernier point par l'inverse de k.

Dans ce cas, plusieurs sous-problèmes se présentent, sur lesquels nous avons enquêté.

cas 1 : Par exemple, on a : $a = 1$, $b = 0$, $z = 13$. L'équation est : $y^2 = x^3 + x \mod 13$. Les points inclus dans cette courbe sont :

(0,0)	(2,6)	(2,7)	(3,2)	(3,11)	(4,4)	(4,9)	(5,0)	(6,1)	(6,12)
(7,5)	(7,8)	(8,0)	(9,6)	(9,7)	(10,3)	(10,10)	(11,4)	(11,9)	

TABLE 5.6 – Les points de courbe

Donc, si nous prenons par exemple un point quelconque de cette courbe, comme (3,11), et que nous le multiplions par 5, cela donne le point (0,0). (3,11) * 5 = (0,0)
Nous avons $k = 5$, et nous devons maintenant rechercher l'inverse de k. (0,0) * ? = (3,11)
Donc, nous avons dit que nous utiliserions la méthode exhaustive d'addition pour trouver l'inverse de k.Pour cet exemple, si nous additionnons le point (0,0) avec lui-même, nous obtenons l'infini, ce qui signifie que le point (0,0) est son propre inverse. Cela nous entraîne dans une boucle infinie, nous empêchant de trouver l'inverse de k dans ce cas. Cet exemple appartient au premier type de courbes elliptiques, mais ce problème peut parfois se poser. La solution consiste à choisir un point de manière à éviter ce problème, ou bien, si ce problème survient lors de la génération de la clé, à changer la valeur de k. Cependant, si ce problème se produit lors du déchiffrement du message, cela pose un problème. C'est

pourquoi il est préférable d'éviter ces courbes et de se baser sur les courbes du deuxième cas.

cas 2 : Ce type de courbe elliptique est le meilleur pour trouver l'inverse_k, car pour inverser un point quelconque sur cette courbe, on passe par tous les autres points de la courbe. Ainsi, la probabilité de rencontrer le problème du premier cas est nulle. En effet, si l'on multiplie le point par n'importe quel nombre, on obtient un point appartenant à cette courbe. De plus, en utilisant la méthode inverse, on évite le problème de la boucle infinie. On a : $a = 2$, $b = 2$, $z = 17$. L'équation est : $y^2 = x^3 + 2x + 2 \mod 17$. Les points inclus dans cette courbe sont :

(0,6)	(0,11)	(3,71)	(3,16)	(5,1)	(5,16)	(6,93)	(6,14)	(7,6)	
(7,11)	(9,1)	(9,16)	(10,6)	(10,11)	(13,7)	(13,10)	(16,4)	(16,13)	

TABLE 5.7 – Les points de courbe

Si nous prenons un point quelconque de cette courbe, comme (3,1), et que nous le multiplions par 5, cela donne le point (5,1). (3,1) * 5 = (5,1)
Nous avons $k = 5$, et nous devons maintenant rechercher l'inverse de k. $(5, 1)*? = (3, 1)$
$(5, 1) + (5, 1) = (6, 3)$
$(6, 3) + (5, 1) = (10, 6)$
$(10, 6) + (5, 1) = (3, 1)$
Donc $\text{Inv}_k = 4$.
En utilisant n'importe quelle valeur de k, nous trouvons une solution correcte.Le problème qui subsiste dans la recherche est de déterminer la relation qui distingue les courbes du premier type des courbes du deuxième type.

5.9 Conclusion

Dans ce chapitre on a pu exposer l'approche proposée pour notre sujet d'étude, en présentant les formules théoriquement prouvées avec les différentes étape et outils requis pour leur implémentation. On a également présenté une analyse des résultats de quelques tests effectués, afin de prouver son exactitude et sa fiabilité.

CONCLUSION GÉNÉRALE

Le chiffrement de Merkle-Hellman enrichi par l'intégration des courbes elliptiques, offre des perspectives prometteuses pour le développement de systèmes cryptographiques plus sécurisés. Bien qu'innovant à son époque il a montré des vulnérabilités face aux avancées en cryptanalyse. En revanche, les courbes elliptiques sont reconnues pour leur robustesse et leur capacité à fournir une sécurité équivalente avec des clés de taille réduite.

Dans ce modeste travail, nous avons abordé une des techniques de la cryptographie représentée par le chiffrement de merkle-helmen en utilisant les courbes elliptiques, Notre mémoire a été élaborée suivant un plan composé en quatre chapitres. Le premier chapitre, consacré quelques techniques de chiffrement de Merkle-hellman. Tandis que dans le deuxième chapitre, nous avions abordé aux bases mathématiques requises explique les différentes opérations sur les courbes elliptiques ainsi que le problème du logarithme discret elliptique, ce qui nous a permis par la suite d'analyser quelques travaux de recherche dans le domaine qui composent notre état de l'art dans un troisième chapitre, grâce auquel on a enfin pu mettre en place notre propre approche dont on a exposé le côté théorique et pratique dans un dernier chapitre, et présenté également une analyse des résultats obtenus.

Cependant, cette amélioration de la sécurité a un coût : une augmentation de la complexité calculatoire. Cela souligne l'importance de trouver un équilibre entre la robustesse de la sécurité et l'efficacité pratique des algorithmes cryptographiques.

En conclusion, l'intégration des courbes elliptiques dans le chiffrement de Merkle-Hellman constitue une avancée notable dans le domaine de la cryptographie asymétrique. Ce travail ouvre la voie à de futures recherches et optimisations, visant à exploiter pleinement le potentiel des courbes elliptiques pour renforcer les systèmes de chiffrement basés sur le problème du sac à dos, tout en maintenant une performance acceptable pour des applications pratiques. La continuité de ces explorations est essentielle pour faire face aux défis croissants en matière de sécurité informatique et de protection des données.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[1] Makni Abdelhakim et Meguenni Khalil. *Étude et implémentation du chiffrement de Merkle Hellman. Mémoire Master. Université Belhadj Bouchaib*. 2017.

[2] "Chiffrement symétrique ou asymétrique : Le guide comparatif ultime". In : (2014). Accessed : 2024-04-04. URL : https://www.ssldragon.com/fr/blog/chiffrement-symetrique-asymetrique/.

[3] Axel Durbet. *Introduction à la cryptologie. Rapport. Université Clermont Auvergne : IUT Aurillac*. 2022.

[4] Clotilde Guillet et Adrianna Yombi. *Le Knapsack et le chiffre de Merkle-Hellman. Rapport. UNIVERSITÉ CLAUDE BERNARD LYON1*. 2008.

[5] Hind Ikni. "Partage de Secrets Cryptographiques en Utilisant les Courbes Elliptiques. Mémoire Master". In : *Université Belhadj Bouchaib* (2018).

[6] Matthieu Legeay. "Loi de groupe sur une courbe elliptique. exposé de séminaire proposés. Université de Rennes". In : (2008-2009).

[7] Ismail Lotfi. "Cryptographie à base de courbes elliptiques". In : *Thèse de Doctorat, Université de technologie-Nanyang* (2017).

[8] Nicolas Méloni. "Arithmetic for Cryptography based on Elliptic Curves". Thèse de doct. Université de Montpellier 2, 2007.

[9] Koichiro Noro et Kunikatsu Kobayashi. "Knapsack Cryptosystem on Elliptic Curves". In : *EPRINT Journal* 6 (2009).

[10] "Qu'est-ce que la cryptographie ?" In : (2018). Accessed : 2024-05-18. URL : https://www.fortinet.com/fr/resources/cyberglossary/what-is-cryptography.

[11] R. Rajaram Ramasamy et al. "Knapsack based ECC encryption and decryption". In : *Int. J. Netw. Secur.* 9.3 (2009), p. 218-226.