République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Centre Universitaire Belhadj Bouchaib d'Aïn-Témouchent

Faculté des Sciences

Département de Mathématiques et de l'Informatique

Mémoire

Pour l'obtention du Diplôme de Master en Informatique

Option : Réseaux et Ingénierie des Données (RID)

Présenté par :

M. Abdelhakim MAKNI & M. Houssin Sabri MEDJAHRI

Cryptographie à Base de Couplage : Chiffrement à Base d'Identité

Encadrant :

M. Hichem Bouchakour Errahmani

Maitre de Conférence "B" à C.U.B.B.A.T.

Soutenu en 2019

Devant le jury composé de :

Président :	M. Fethi Imad BENARIBI (M.A.A)	C.U.B.B.A.T.
Examinateur :	M. Djalal MERAD BOUDIA (M.A.A)	C.U.B.B.A.T.
Encadrant :	M. Hichem Bouchakour Errahmani (M.C.B)	C.U.B.B.A.T.

Remerciements

Alhamdulillah, notre source de force et de patience durant toutes ces longues années.

Toute notre reconnaissance va en premier lieu à notre encadrant sans qui ce travail n'aurait pas été possible, monsieur Hichem BOUCHAKOUR ERRAHMANI, qui a mis en toute modestie son temps, ses conseils à notre service et a su comment nous aidé à façonner et mettre en concrétisation nos idées.

Notre plus profonde gratitude va à nos chères parents, qui nous ont soutenu et mis à notre disposition toutes les chances pour réussir et cela tout au long de notre cursus. On espère que vous trouverez à travers ce travail si ce n'est qu'une infime partie de l'aboutissement de vos efforts.

On serait méconnaissant si on passait sans remercier nos camarades et amis pour leurs soutien moral et leurs encouragements, plus particulièrement Amine, Djamel Eddine, Fethellah et Omar.

Enfin on voudrait remercier les membres de juré monsieur Fethi Imad BENARIBI et monsieur Djalal MERAD BOUDIA de l'intérêt porté à notre travail, ainsi que l'ensemble des enseignants du département qui ont enrichi nos connaissances et nous ont guidé tout au long de notre cursus.

Résumé

Contrairement au chiffrement symétrique, le chiffrement asymétrique permet d'envoyer des messages sans établir de secret au préalable, grâce à une autorité de certification de clé publique. Le chiffrement à base d'identité introduit par Shamir en 1984, nous dispense de cette autorité en construisant la clé publique à partir de l'identité du récepteur. Dans ce travail, nous présentons un schéma de chiffrement à base d'identité inspiré de celui de Wee, utilisant les courbes elliptiques et la notion du couplage qui est une application bilinéaire entre des groupes construits utilisant ces courbes. En plus d'explorer les différentes notions qui englobe notre thématique intitulé *« Cryptographie à base de couplage : Chiffrement à base d'identité »* tel que les courbes elliptiques, le couplage et le système de chiffrement à base d'identité, on présente quelques travaux faits dans ce domaine et qui nous ont permis de construire notre propre système et d'obtenir de bons résultats avec l'outil de programmation Python. Comme futur travaux, on espère enrichir plus notre propre bibliothèque de couplage ainsi qu'intégrer d'un schéma de signature numérique pour renforcer la sécurité de notre schéma.

Mots clés : Cryptographie, courbe elliptique, couplage, chiffrement à base d'identité

Abstract

Unlike the symmetric encryption, asymmetric encryption allows us to send messages without establishing a secrecy in the advance, thanks to a public key certification authority. The identity-based encryption introduced by Shamir in 1984, exempts this authority by constructing the public key from the identity of the receiver. In this work we present an identity-based encryption scheme inspired from the one of WEE, using elliptic curves and the concept of pairing which is a bilinear map between groups constucted using these curves. Besides of exploring the various concepts that encompasses our theme entitled *"Pairing-based cryptography : identity-based encryption"* such as elliptic curves, pairing and the identity-based encryption system, we present some of the work done in this domain and which have allowed us to build our own system and obtain good results using Python. As future work, we hope to enrich our own pairing library as well as integring a scheme of digital signature to strengthen the security of our scheme.

Keywords: Cryptography, elliptic curve, pairing, identity-based encryption

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Chapitre 1

Introduction générale

Pour des fins militaires et diplomatiques, l'humain avait besoin des techniques pour dissimuler les informations cruciales lors des échanges de messages, en rendant illisibles ces derniers à toute autre partie n'étant pas censé être en possession de ces messages.

Au fil du temps, la cryptologie et ses branches (cryptographie et cryptanalyse) n'ont pas cessé d'évoluer, allant de la Scytale spartiate, passant par le fameux chiffrement de César, jusqu'à la célèbre machine Enigma conçue par les allemands, ils ont tous la même particularité, où la même clé est utilisée pour chiffrer et déchiffrer un message (cryptographie symétrique). Arrivant à des cryptosystèmes qui utilisent la notion de clé publique (cryptographie asymétrique), une solution au problème d'échange de clé avant même d'établir la communication, oú un schéma de partage de clé et d'authentification est établi. Ces systèmes sont fondés sur des problèmes mathématiques connus comme étant difficiles, tel que la factorisation des entiers pour le système RSA [1], ou bien le problème du logarithme discret pour Diffie-Hellman [2] et ElGamal [3].

Cependant, la croissance et le progrès dans les domaines de la cryptanalyse et de l'algorithmique ont fait que certaines notions deviennent vulnérables ou obsolètes carrément, rendant ainsi la résolution des problèmes difficiles possible en un temps polynomial. Puis, étant donné l'avancée des réseaux internet et l'implication de la sécurité numérique, de nombreuses questions nous font face :

- Peut-on amener ces problèmes mathématiques, à des niveaux de sécurité décents et surpassant les technologies d'aujourd'hui ?
- Est-il possible d'utiliser les notions de la cryptanalyse pour développer celles de la cryptographie ?

- Éxiste-t-il un schéma plus simple pour la génération de la clé publique ?
- Comment mettre en concrétisation un schéma pouvant répondre à ces question ?

En se basant sur l'ensemble varié et large des documents dédiés à ce sujet, nous avons pu espérer donner des réponses pertinentes et factuelles, en allant au-delà d'un simple cryptosystème à clé publique.

En effet, avec une projection du problème de Diffie-Hellman sur les courbes elliptiques rendant ainsi la résolution du problème plus complexe. Ainsi l'utilisation du couplage, initialement utilisé pour de mauvaises intentions, puis exploité par Joux [4] pour des fins Cryptographiques, le couplage permet d'accroître cette complexité. Enfin l'utilisation d'un schéma proposé par Shamir [5] en 1984, intitulé schéma de chiffrement à base d'identité, nous dispose de la mise en place d'une infrastructure de gestion de clés publiques, oú nous avons juste besoin d'une information sur l'identité du destinataire.

Afin de ne pas manquer à la bonne présentation de notre sujet intitulé *« Cryptographie à base de couplage : Chiffrement à base d'identité »*, nous avons structuré notre mémoire de la façon suivante :

Le premier chapitre est consacré aux généralités mathématiques qui englobe la thématique des courbes elliptiques, oú des rappels sont nécessaires, avant de nous approfondir sur les opérations dans les courbes elliptiques et nous parlerons ensuite du problème de Diffie-Hellman et sa projection sur les courbes elliptiques.

Ensuite, nous entamons la notion de couplage, en donnant une définition générale sur ce qu'est le couplage, ainsi que ces types. Nous décrivons en plus des notion qui lui sont relatives et tous ce qui concerne son calcul. Une partie est consacré pour citer quelques unes de ses applications.

Dans le troisième chapitre, il sera question des schémas de chiffrement à base d'identité, en accordant une partie à la définition traditionnelle d'un schéma de chiffrement à base d'identité. Nous parlerons aussi dans ce chapitre de la notion de la sécurité sémantique dans ce genre de schéma, ce qui nous permettra d'étaler une discussion sur les travaux effectués dans ce domaine, et d'établir une synthèse.

Par la suite, nous présentons notre contribution, nous décrivons aussi notre implémentation du couplage et la concrétisation de l'approche proposée et nous donnons un exemple de son fonctionnement.

On clôture notre travail par une conclusion où on parle de notre contribution et des résultats obtenus ainsi que nos perspectives pour les futurs travaux.

Chapitre 2

Courbes elliptiques

Sommaire

2.1 Introduction

L'explosion de la taille des clés utilisées dans les algorithmes classiques de la cryptographie à clé publique tel que l'échange de clé de Diffie-Hellman [2], le chiffrement RSA [1] et le chiffrement ELGAMAL [3] pour assurer leurs sécurité a nécessité la recherche d'autres solutions alternatives, qui utilisent des clés plus petites et qui fournissent le même niveau de sécurité, en outre, la cryptographie par les courbes elliptiques a vu le jour. Ces dernières sont des primitives mathématiques utilisées dans des domaines différents, dans la cryptographie elles sont utilisées pour construire des systèmes de sécurité complexes tel que les systèmes d'échange de clés, la cryptographie à clé publique, la signature numérique, les générateurs de nombres pseudo-aléatoires...etc, nous présentons dans ce chapitre les courbes elliptiques et nous donnons quelques unes de leurs applications dans la cryptographie.

2.2 Rappels mathématiques

Dans la cryptographie, on s'intéresse aux courbes elliptiques définies sur des corps finis, ces derniers sont des structures algébrique particulières composées d'un nombre fini d'éléments qu'on peut additionner, soustraire, multiplier et diviser, cette section est consacrée à la définition des corps finis ainsi que d'autres structures algébriques que nous allons utiliser après, tel que les groupes, les groupes cycliques et les extensions de corps finis.

2.2.1 Groupe

Définition 2.2.1 (Groupe)**.** Un groupe est un ensemble d'éléments $\mathbb{G}$ muni d'une opération binaire $\circ$, noté $(\mathbb{G}, \circ)$ ou simplement $\mathbb{G}$ et qui doit satisfaire les conditions suivantes :

1. L'opération $\circ$ est une loi interne sur $\mathbb{G}$. $\forall x, y \in \mathbb{G}, x \circ y = z \in \mathbb{G}$
2. L'opération $\circ$ est associative. $\forall x, y, z \in \mathbb{G}, (x \circ y) \circ z = x \circ (y \circ z)$
3. Il existe un élément neutre $e \in \mathbb{G}$ tel que pour tout $x \in \mathbb{G}, x \circ e = e \circ x = x$
4. Pour tout $x \in \mathbb{G}$ il existe son inverse $x^{-1} \in \mathbb{G}$ tel que $x \circ x^{-1} = x^{-1} \circ x = e$
5. **(Optionnelle)** pour être considéré comme un groupe abélien (ou commutatif), l'opération $\circ$ doit être commutative. $\forall x, y \in \mathbb{G}, x \circ y = y \circ x$

Définition 2.2.2 (Ordre de groupe)**.** Un groupe est fini, s'il contient un nombre fini d'éléments, ce nombre présente la cardinalité ou l'ordre de groupe et il est noté

$|\mathbb{G}|$ ou $\#\mathbb{G}$.

2.2.2 Groupe cyclique

Définition 2.2.3 (Ordre d'un élément de groupe)**.** L'ordre d'un élément x noté $ord(x)$ dans un groupe est le plus petit entier positif q tel que $x^q = \underbrace{x \circ x \circ ... \circ x}_{q\ fois} = e$ avec e l'élément neutre du groupe.

Définition 2.2.4 (groupe cyclique)**.** Un groupe $(\mathbb{G}, \circ)$ est cyclique, s'il contient un élément a tel que $ord(a) = |G|$, autrement dit, chaque élément x du groupe peut s'écrire sous la forme $x = a^i$ pour tout i dans le groupe, a est appelé l'élément primitif ou générateur du groupe et on note $G = \langle a \rangle$.

Théorème 2.2.1 (sous groupe cyclique)**.** *Soit $(\mathbb{G}, \circ)$ un groupe cyclique, alors tout élément $a \in \mathbb{G}$ avec $ord(a) = S$, est un générateur du sous groupe cyclique qui contient S élément [6].*

Remarque. Dans tout ce que suit, nous allons utiliser les termes groupe additif et groupe multiplicatif pour respectivement le groupe muni de l'opération d'addition $+$ dont l'élément neutre est le 0, et le groupe muni de l'opération de multiplication $\times$ dont l'élément neutre est le 1.

2.2.3 Corps

Définition 2.2.5 (corps)**.** Un corps $\mathbb{F}$ est un ensemble d'éléments muni des opérations d'addition et de multiplication et qui doit satisfaire les conditions suivantes :

1. Tous les éléments de $\mathbb{F}$ forment un groupe additif muni de l'opération d'addition $+$ dont l'élément neutre est le 0.
2. Tous les éléments de $\mathbb{F}$ à l'exception du 0 forment un groupe multiplicatif muni de l'opération de multiplication $\times$ dont l'élément neutre est le 1.
3. l'opération $\times$ est distributive par rapport à l'opération $+$. $\forall x, y, z \in \mathbb{F}, x(y + z) = xy + xz$

2.2.4 Corps finis

Théorème 2.2.2 (Existence d'un corps fini)**.** *Un corps $\mathbb{F}$ avec l'ordre m existe seulement si m est la puissance d'un nombre premier p, $m = p^n$ avec n un entier positif, on appelle $\mathbb{F}_{p^n}$ un corps fini et p et sa caractéristique [6].*

2.2.5 Extensions de corps finis

Le calcul du couplage présenté dans le chapitr 3 repose beaucoup sur une arithmétique sur des extensions de corps $\mathbb{F}_{p^k}$ avec k un entier appelé le degré de plongement, une arithmétique qui est beaucoup plus coûteuse qu'une arithmétique sur des corps finis simples de la forme $\mathbb{F}_p$.

La représentation naturelle d'un élément qui appartient à $\mathbb{F}_{p^k}$ est un polynôme de degré $k-1$ et ses coefficients sont dans $\mathbb{F}_p$ comme suit :

$$\mathbb{F}_{p^k} = \{A(x) \; tel \; que \; A(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + ... + a_1x^1 + a_0 \; avec \; a_i \in \mathbb{F}_p\}$$

L'addition et la soustraction dans $\mathbb{F}_{p^k}$ sont des additions et des soustractions naturelles des polynomiaux, de même pour la multiplication et la division sauf que le résultat de ces deux derniers doit avoir une opération modulo un polynôme irréductible $f(x)$ de degré k et aussi ses coefficients sont dans $\mathbb{F}_p$, sachant qu'un polynôme irréductible peut être à-peut-prés comparé à un nombre premier, leurs diviseurs sont seulement 1 et eux mêmes, ce polynôme irréductible doit être choisi soigneusement pour minimiser le calcul, généralement il est choisi de la forme $x^k - \alpha$ avec α une constante non-cube et non-carrés dans $\mathbb{F}_p$ puisqu'il est démontré qu'un polynôme de cette forme est irréductible sous certaines conditions (Théorème 1 dans [7]) et aussi il permet de minimiser le calcul, en effet on utilise généralement la notation $\mathbb{F}_p[x]/f(x)\mathbb{F}_p[x]$ pour une extension de corps.

Cette représentation naturelle d'un élément de $\mathbb{F}_{p^k}$ reste efficace pour $k = 2$ ou $k = 3$ où on a pas un autre choix, mais pour $k > 3$ il est obligatoire de concevoir toute une nouvelle implémentation optimale de $\mathbb{F}_{p^k}$ pour chaque k qu'on veut travailler avec, il existe une alternative qui parait beaucoup plus rapide et pratique, cette dernière a été proposée pour la première fois par Baktir et Sunar [8] qui ont utilisé des degrés de plongement de la forme $k = 2^i3^j$ et ont construit le corps $\mathbb{F}_{p^k}$ correspondant en utilisant une série successive d'extensions quadratiques et cubiques, pour présenter par exemple une extension de corps $\mathbb{F}_{p^{12}}$ on a le choix d'utiliser :

- une extension cubique d'une extension quadratique de $\mathbb{F}_{p^2}$: cas 2,2,3 où $\mathbb{F}_{p^{12}}$ se voit comme $\mathbb{F}_{\left((p^2)^2\right)^3}$.
- une extension quadratique d'une extension cubique de $\mathbb{F}_{p^2}$: cas 2,3,2 où $\mathbb{F}_{p^{12}}$ se voit comme $\mathbb{F}_{\left((p^2)^3\right)^2}$.
- une extension quadratique d'une autre extension quadratique de $\mathbb{F}_{p^3}$; cas 3,2,2 où $\mathbb{F}_{p^{12}}$ se voit comme $\mathbb{F}_{\left((p^3)^2\right)^2}$.

Dans chaque extension quadratique ou cubique un polynôme de la forme $x^k - \alpha$ est choisi, si on prend par exemple le deuxième cas de l'exemple précédent, la construction peut être schématisée comme suit :

$$\mathbb{F}_p \xrightarrow{\beta^2-\alpha} \mathbb{F}_{p^2} \xrightarrow{\gamma^3-\beta} \mathbb{F}_{p^6} \xrightarrow{\delta^2-\gamma} \mathbb{F}_{p^{12}}$$

Cette technique de construction de l'extension de corps est appelée la tour d'extension de corps. On continue avec le même cas du même exemple, un élément de $\mathbb{F}_{p^{12}}$ est alors de la forme $a = a_0 + a_1\delta$ avec a_0 et $a_1 \in \mathbb{F}_{p^6}$, donc chaque opération dans $\mathbb{F}_{p^{12}}$ tourne vers des opérations dans $\mathbb{F}_{p^6}$, de même un élément de $\mathbb{F}_{p^6}$ est alors de la forme $b = b_0 + b_1\gamma + b_2\gamma^2$ avec b_0, b_1 et $b_2 \in \mathbb{F}_{p^2}$ et chaque opération dans cette extension tourne vers des opérations dans $\mathbb{F}_{p^2}$, de cette façon une opération descend la tour jusqu'à arriver à des opérations de base dans $\mathbb{F}_p$.

La tour d'extension de corps est favorisée par rapport à la représentation naturelle de l'extension de corps, car il existes plusieurs algorithmes efficaces de l'arithmétique pour des extensions de corps de la forme $\mathbb{F}_{p^2}$ et $\mathbb{F}_{p^3}$, pour plus de détails sur l'arithmétique des extensions quadratiques et cubiques des corps on recommande au lecteur de voir le chapitre 5 de [9].

2.3 Courbes elliptiques sur $\mathbb{R}$

Une courbe elliptique sur l'ensemble des nombres réels $\mathbb{R}$ est une courbe sur le plan définie par l'équation dite de Weierstrass suivante :

$$y^2 = x^3 + ax + b \text{ avec } x, y, a, b \in \mathbb{R}$$

Les coefficients a et b déterminent la forme de la courbe, comme on le voit dans la figure 2.1.

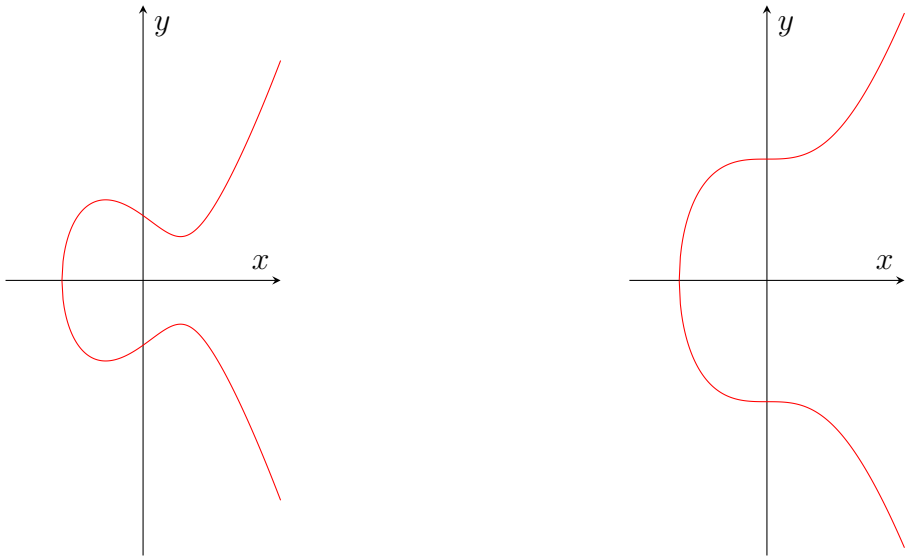

Figure 2.1 – Exemples des courbes elliptiques sur $\mathbb{R}$

La définition nécessite aussi que la condition suivante : $4a^3 + 27b^2 \neq 0$ soit vérifiée pour assurer que la courbe soit non-singulière ou lisse, autrement dit, l'équation $x^3 + ax + b = 0$ admet des solutions distinctes. Géométriquement, cela se traduit par le fait que chaque point dans la courbe n'est pas un point double, ni point de rebroussement et admet une tangente bien définie.

On remarque que la courbe est symétrique par rapport à l'axe des abscisses.

Une courbe elliptique $E(\mathbb{R})$ est constituée de l'ensemble des points dont leurs cordonnées (x, y) satisfont l'équation de la courbe, avec un point $\mathcal{O}$ dit le point à l'infinie ou le point idéal.

$$E(\mathbb{R}) = \{(x, y) \in \mathbb{R}^2 : y^2 = x^3 + ax + b \text{ avec } a, b \in \mathbb{R}\} \cup \{\mathcal{O}\}$$

Le point à l'infinie $\mathcal{O}$ présente l'élément neutre de l'opération de l'addition géométrique de points dans la courbe, nous allons fournir plus de détails concernant cette dernière ultérieurement.

2.4 Courbes elliptiques sur $\mathbb{F}_p$

2.4.1 Définition

Soit $x, y, a, b \in \mathbb{F}_p$ avec p premier et $4a^3 + 27b^2 \neq 0$, une courbe elliptique sur un corps fini $\mathbb{F}_p$ est définie par l'équation :

$$y^2 = x^3 + ax + b$$

Ainsi toutes les opérations arithmétiques se font dans $\mathbb{F}_p$.

On note $E/\mathbb{F}_p$ une courbe E définie sur le corps fini $\mathbb{F}_p$ et $E(\mathbb{F}_p)$ l'ensemble des points dont leurs cordonnées (x, y) satisfont l'équation de la courbe, avec un point $\mathcal{O}$ dit le point à l'infinie.

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p^2 : y^2 = x^3 + ax + b \text{ avec } a, b \in \mathbb{F}_p\} \cup \{\mathcal{O}\}$$

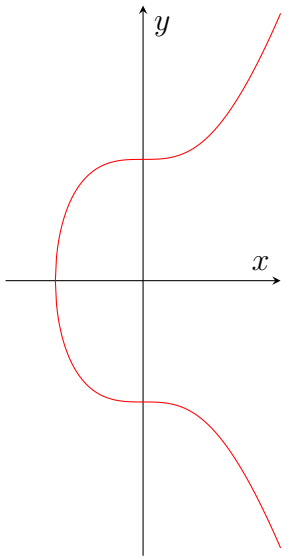

Figure 2.2 – Courbe elliptique définie sur $\mathbb{R}$ par l'équation $y^2 = x^3 + 7$

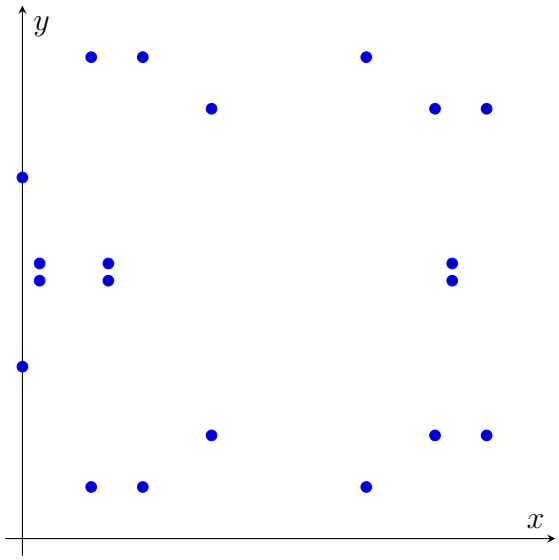

Figure 2.3 – Courbe elliptique définie sur $\mathbb{F}_{31}$ par l'équation $y^2 = x^3 + 7$

Comme le montre la figure 2.3 le graphe d'une courbe elliptique définie sur un corps fini $\mathbb{F}_p$ n'est pas pratiquement lisible, c'est pour ça que dans tout ce qui suit nous allons utiliser le graphe de l'équation sur l'ensemble des nombres réels $\mathbb{R}$ correspondante, ce qu'on le voit dans la figure 2.2 .

2.4.2 Nombre de points dans une courbe elliptique

Nous allons voir après la démonstration que l'ensemble des points sur une courbe elliptique forme un groupe commutatif, or, il est maintenant très important de déterminer le nombre des points dans ce groupe. Ce nombre nous l'avons déjà nommé l'ordre de groupe noté $|E(\mathbb{F}_p)|$ ou $\#E(\mathbb{F}_p)$.

Le théorème de Hass suivant permet de déterminer approximativement l'ordre $\#E(\mathbb{F}_p)$ de groupe $E(\mathbb{F}_p)$.

Théorème 2.4.1 (Théorème de Hass)**.** *soit $E/\mathbb{F}_p$ une courbe elliptique définie sur un corps fini, alors :*

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

Il est important de remarquer que pour p très grand on obtient $\#E(\mathbb{F}_p) \approx p$ [6] .

2.4.3 Addition géométrique de points

Pour que l'ensemble des points dans une courbe elliptique forme un groupe commutatif, une opération de groupe est requise, cette opération est représentée par le symbole de l'addition + et n'a aucune relation avec l'addition habituelle des nombres.

L'opération de l'addition de points exploite deux propriétés des courbes elliptiques :

- Chaque ligne qui passe par deux points dans une courbe elliptique doit intersecter cette courbe dans un troisième point, de fait que l'équation de la courbe est de degré 3 et celle de la ligne est de degré 1 (théorème de Bezout).
- Pour tout point muni des cordonnées (x, y) dans une courbe elliptique alors le point muni des cordonnées $(x, -y)$ est aussi dans la courbe de faite que cette dernière est symétrique par rapport à l'axe des abscisses.

Soit $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ deux point dans $E(\mathbb{F}_p)$, on note l'addition de ces deux points $P + Q = R$ tel que $R = (x_R, y_R)$ est un autre point dans $E(\mathbb{F}_p)$.

L'élément neutre de cette opération est le point à l'infinie $\mathcal{O}$, et l'élément inverse d'un point $P = (x_P, y_P)$ est $-P = (x_P, -y_P)$. Par conséquent on obtient les propriétés suivantes :

- $P + \mathcal{O} = \mathcal{O} + P = P$
- $P + (-P) = -P + P = \mathcal{O}$

Soit $E/\mathbb{F}_p$ une courbe elliptique définie sur un corps fini et $P, Q \in E(\mathbb{F}_p)$, on distingue deux cas où l'interprétation géométrique de l'addition des points est définie comme suit :

Addition des points $R = P + Q$ avec $P \neq Q$: on dessine la ligne L qui passe par P et Q et qui intersecte la courbe $E/\mathbb{F}_p$ dans un troisième point noté R', on dessine en suite la ligne verticale L' qui passe par R' et qui intersecte la courbe dans un point noté $R = P + Q$ qui est le résultat de l'addition de P et Q. (Voir la figure 2.4)

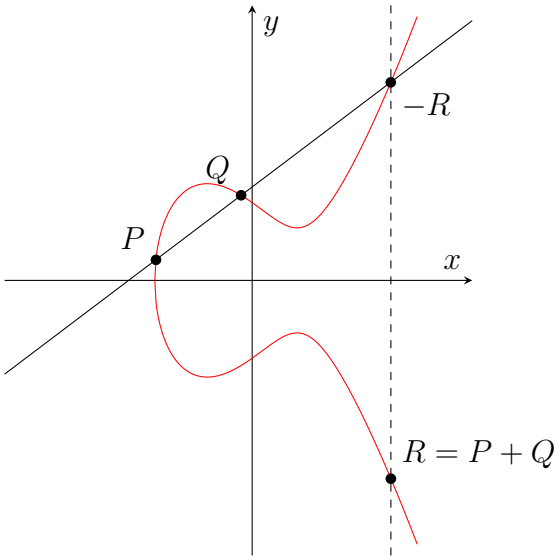

Figure 2.4 – Addition des points

Doublement de point $R = P + P = 2P$: on dessine la tangente L de la courbe qui passe par P et qui intersecte la courbe $E/\mathbb{F}_p$ dans un autre point noté R', on dessine en suite la ligne verticale L' qui passe par R' et qui intersecte la courbe dans un point noté $R = P + P = 2P$, qui est le résultat du doublement de P. (Voir la figure 2.5)

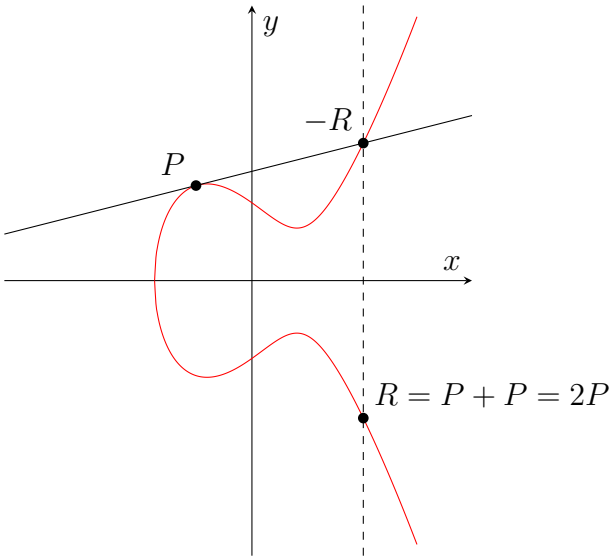

Figure 2.5 – Doublement de point

De cette interprétation géométrique (les deux cas), des formules pour calculer l'addition $R = P + Q$ peuvent être dérivées depuis l'intersection de la courbe $E/\mathbb{F}_p$: $y^2 = x^3 + ax + b$ et la ligne $L : y = Sx + V$ comme suit :

$$x_R = S^2 - x_P - x_Q$$

$$y_R = S(x_P - x_R) - y_P$$

$$\text{avec } S = \begin{cases} \dfrac{y_P - y_Q}{x_P - x_Q} \text{ Si } P \neq Q \\ \dfrac{3x_P^2 + a}{2y_P} \text{ Si } P = Q \end{cases}$$

Toutes les opérations arithmétique se font dans $\mathbb{F}_p$.

Il est maintenant facile de démontrer que l'ensemble des points dans une courbe elliptique avec le point à l'infinie $\mathcal{O}$ muni de l'opération d'addition de points forme un groupe de faite qu'on a déjà démontrer que le résultat de l'addition est un point de la courbe, de plus il a déjà été dit que l'élément neutre est le point à l'infinie $\mathcal{O}$ et que l'élément inverse d'un point $P = (x_P, y_P)$ est $-P = (x_P, -y_P)$, il reste à démontrer l'associativité, cette dernière peut être démontrée en calculant les deux cotés de l'équation $(A + B) + C = A + (B + C)$, de plus ce groupe de points est abélien puisqu'il est claire que l'addition de points est commutative (la ligne qui passe par P et Q est la même ligne qui passe par Q et P).

Dans la cryptographie on s'intéresse beaucoup plus aux groupes cycliques, sachant comment les utiliser pour concevoir des cryptosystèmes. Maintenant on a tous les concepts essentiels pour présenté le théorème suivant :

Théorème 2.4.2. *Les points dans une courbe elliptique avec le point à l'infinie $\mathcal{O}$ ont des sous groupes cycliques. Sous certaines conditions, tous les points dans une courbe elliptique forment un groupe cyclique [6].*

2.4.4 Systèmes de coordonnées

Nous avons vue que les formules d'addition et de doublement de point nécessitent une division (ou inversion), un calcul qui est beaucoup plus complexe que la multiplication dans un corps. Par conséquent, l'utilisation d'autres systèmes de cordonnées comme les coordonnées projectives ou Jacobiennes est nettement conseillée pour éviter l'inversion et donc diminuer la complexité des opérations de l'addition et de doublement de points.

Coordonnées projectives

Les coordonnées projectives sont représentés par un triplet (X, Y, Z), avec $Z \neq 0$ où $(X/Z, Y/Z)$ est le point affine correspondant. Ces coordonnées sont homogènes c'est-à-dire que pour tout $\alpha \neq 0$, le point $(\alpha X, \alpha Y, \alpha Z)$ est équivalent au point

(X, Y, Z).
L'équation d'une courbe elliptique sur un corps fini $\mathbb{F}_p$ utilisant des coordonnées projectives devient alors :

$$E(F_p) = (X, Y, Z) \in F_p^3, tel\ que\ Y^2Z = X^3 + aXZ^2 + bZ^3$$

Les coordonnées du point à l'infini $\mathcal{O} = (0, 1, 0)$ et l'opposé d'un point $P = (X_P, Y_P, Z_P)$ est $-P = (X_P, -Y_P, Z_P)$

Addition Soient $P = (X_P, Y_P, Z_P)$ et $Q = (X_Q, Y_Q, Z_Q)$ deux points donnés en coordonnées projectives de la courbe et soient $A = Y_QZ_P - Y_PZ_Q$, $B = X_QZ_P - X_PZ_Q$ et $C = A^2Z_PZ_Q - B^3 - 2B^2X_PZ_Q$
Les cordonnées du point R qui est la somme de P et Q sont obtenues comme suit :

$X_R = BC$
$Y_R = A(B^2X_PZ_Q) - C - B^3Y_PZ_Q$
$Z_R = B^3Z_PZ_Q$

Doublement Soit $P = (X_P, Y_P, Z_P)$ un point de la courbe en coordonnés projectives et soient $A = aZ_P^2 + 3X_P^2, B = Y_PZ_P, C = X_PY_PB$, et $D = A^2 - 8C$
Les cordonnées du point R qui est le double du point P sont obtenues comme suit :

$X_R = 2BD,$
$Y_R = A(4C - D) - 8Y_P^2B^2,$
$Z_R = 8B^3.$

Coordonnées Jacobiennes

Les coordonnées Jacobiennes sont représentés par un triplet (X, Y, Z), avec $Z \neq 0$ où $(X_P/Z_P^2, Y_P/Z_P^3)$ est le point affine correspondant. Ces coordonnées sont aussi homogènes.
L'équation d'une courbe elliptique sur un corps fini $\mathbb{F}_p$ utilisant des coordonnées Jacobiennes devient alors :

$$E(F_P) = (X, Y, Z) \in F_p^3, \text{ tel que } Y^2 = X^3 + aXZ^4 + bZ^6$$

Les coordonnées du point à l'infini $\mathcal{O} = (1, 1, 0)$ et l'opposé d'un point $P = (X_P, Y_P, Z_P)$ est $-P = (X_P, -Y_P, Z_P)$

Addition Soient $P = (X_P, Y_P, Z_P)$ et $Q = (X_Q, Y_Q, Z_Q)$ deux points donnés en coordonnées Jacobiennes de la courbe et soient $A = X_PZ_Q^2$, $B = X_QZ_P^2$, $C = Y_PZ_Q^3$, $D = Y_QZ_P^3$, $E = B - A$ et $F = D - C$

Les cordonnées du point R qui est la somme de P et Q sont obtenues comme suit :

$X_R = -E^2 - 2AE^2 + F^2$
$Y_R = -CE^3 + F(AE^2 - X_R)$
$Z_R = EZ_PZ_Q$

Doublement Soit $P = (X_P, Y_P, Z_P)$ un point de la courbe en coordonnés Jacobiennes et soient $A = 4X_PY_P^2$ et $B = 3X_P^2 + aZ_P^4$
Les cordonnées du point R qui est le double du point P sont obtenues comme suit :

$X_R = -2A + B^2$
$Y_R = -8Y_P^4 + B(A - X_R)$
$Z_R = 2Y_PZ_P$

Comparaison entre les systèmes de coordonnées

Le tableau 2.1 présente une comparaison entres les différents systèmes de coordonnées en terme de nombre d'opérations arithmétiques, nous désigneront par S une élévation au carré, par M une multiplication et par I une inversion, ces opérations sont toutes dans un corps et on ne prends pas en considération les additions et les soustractions.

Système de coordonnées	Doublement	Addition
Affine	I + 2M + 2S	I + 2M + S
Projectives	7M + 5S	12M + 2S
Jacobiennes	4M + 6S	12M + 4S

Table 2.1 – Coûts des différents systèmes des coordonnées

En effet une opération d'inversion coute plusieurs opérations de multiplication, cela favorise théoriquement l'utilisation des coordonnées projectives et Jacobiennes par rapport aux coordonnées affines.

2.4.5 Multiplication d'un point par un scalaire

Étant donné un point $P \in E(\mathbb{F}_p)$ et un entier $k \in \mathbb{F}_p$, la multiplication de P par un scalaire k notée kP est définie comme suit :

$$kP = \underbrace{P + P + P + ... + P}_{k\ fois}$$

La méthode naïve pour calculer la multiplication de point P par le scalaire k est l'ajout de P à soit même k fois, celle ci nécessite approximativement 2^m opérations d'addition avec m le nombre de bits de k, cela est pratiquement impossible dans un temps raisonnable pour k très grand. Pour remédier à ce problème plusieurs algorithmes ont été conçus pour calculer cette multiplication, on présente quelque de ces algorithmes.

Double and Add : Cette méthode est basée sur la représentation binaire de l'entier k, on rappelle que cette dernière contient en moyenne $m/2$ coefficients qui sont égaux à 1, en effet elle coute m doublement de points et $m/2$ addition de points, ce qui fait en total $3m/2$ opération [10]. l'algorithme 2.1 présente la méthode Double and Add.

Algorithme 2.1 Double and Add

Entrée: P : point de la courbe elliptique, $k = (k[0], k[1], ..., k[m])_2$: entier
Sortie: $Q = kP$: point de la courbe elliptique
Début
1: $Q \leftarrow P$;
2: **Pour** i de 1 à m **Faire**
3: $Q \leftarrow Q + Q$;
4: **Si** $k[i] = 1$ **Alors**
5: $Q \leftarrow Q + P$;
6: **Fin Si**
7: **Fin Pour**
8: **Retourner** Q ;
Fin

Addition-Subtraction : Cette méthode est basée sur la représentation dite NAF (pour NonAdjacent Form) de l'entier k, qui est une représentation binaire signée c-à-d que ses coefficients appartiennent à $\{-1, 0, 1\}$ avec la propriété qu'il n'y a pas deux coefficients consécutifs différents de 0. Similairement à la forme binaire classique unique d'un entier, il est démontré qu'il existe une représentation NAF unique pour chaque entier, on note que la représentation NAF a le plus grand nombre de 0 parmi toutes les représentations binaires d'un entier (en moyenne seulement $m/3$ des coefficients qui sont pas égaux à 0). On présente dans l'algorithme 2.2 l'une des méthodes pour construire la représentation NAF correspondante à un entier.

Algorithme 2.2 NAF

Entrée: k : entier
Sortie: $k_{NAF}[0], k_{NAF}[1], ..., k_{NAF}[m]$: format NAF d'un entier
Début
1: $i \leftarrow 0$:entier ;
2: **Tant que** $k > 0$ **Faire**
3: **Si** k est impair **Alors**
4: $k_{NAF}[m-i] \leftarrow 2 - (k \bmod 4)$;
5: $k \leftarrow k - k_{NAF}[m-i]$;
6: **Si non**
7: $k_{NAF}[m-i] \leftarrow 0$;
8: **Fin Si**
9: $k \leftarrow k \ / \ 2$;
10: $i \leftarrow i + 1$;
11: **Fin Tant que**
12: **Retourner** $k_{NAF}[0], k_{NAF}[1], ..., k_{NAF}[m]$;
Fin

La méthode Addition-Subtraction présentée dans l'algorithme 2.3 coute approximativement m doublement de points, et $m/3$ addition de points, ce qui fait en total $4m/3$ opération [10], cela favorise théoriquement la méthode Addition-Subtraction de la méthode Double and Add.

Algorithme 2.3 Addition-Subtraction

Entrée: P : point de la courbe elliptique, $k = (k[0], k[1], ..., k[m])_{NAF}$: entier
Sortie: $Q = kP$: point de la courbe elliptique
Début
1: $Q \leftarrow P$;
2: **Pour** i de 0 à $m-1$ **Faire**
3: $Q \leftarrow Q + Q$;
4: **Si** $k[i] = 1$ **Alors**
5: $Q \leftarrow Q + P$;
6: **Fin Si**
7: **Si** $k[i] = -1$ **Alors**
8: $Q \leftarrow Q - P$;
9: **Fin Si**
10: **Fin Pour**
11: **Retourner** Q ;
Fin

2.5 Problème du logarithme discret

Le problème du logarithme discret (DLP pour Discrete-Logarithm Problem) est l'un des premiers problèmes utilisés dans la cryptographie à clé publique, plusieurs

systèmes de sécurité sont conçus à sa base tel que l'échange de clé de Diffie-Hellman [2] et le cryptosystème d'ELGAMAL [3]. DLP était défini la première fois dans un groupe multiplicatif cyclique $\mathbb{Z}_p^*$ des entiers modulo un premier p, mais cela a nécessité que ce p soit très grand ($\geq$ 2048-bit) pour assurer la sécurité des systèmes correspondants. D'une autre part, le groupe des points dans une courbe elliptique définie sur un corps fini était l'une des solutions pour être utilisé avec DLP.

2.5.1 Problème du logarithme discret classique

Définition 2.5.1 (Problème du logarithme discret classique)**.** Soit un groupe multiplicatif cyclique $\mathbb{Z}_p^*$ d'ordre $p-1$, et $a, b \in \mathbb{Z}_p^*$, DLP est le problème de trouver $x \in \mathbb{Z}_p^*$ sachant que $a^x = b \bmod p$.
On appelle x le logarithme discret de b avec la base a et on note $log_a(b)$.

En effet le calcul du logarithme discret est très difficile si les nombres (paramètres) sont assez larges et soigneusement choisis. En dépit de la facilité du calcul de $a^x = b \bmod p$, aller dans le sens inverse est compliqué avec ces conditions, rendant ainsi la fonction une fonction à sens unique.

2.5.2 Problème du logarithme discret dans les courbes elliptiques

Le principe du problème du logarithme discret dans les courbes elliptiques qu'on appellera dès à présent ECDLP (Elliptic Curve Discrete Logarithm Problem) est approximativement le même que celui du DLP.

Définition 2.5.2 (Problème du logarithme discret dans les courbes elliptiques)**.** Soit P et T deux points d'une courbe elliptique $E/\mathbb{F}_p$, résoudre un ECDLP revient à trouver un entier d, où $1 \leq d \leq \#E(\mathbb{F}_p)$ tel que $\underbrace{P + P + P + ... + P}_{d\ fois} = dP = T$.

P est généralement un point avec un grand ordre, et on l'appelle générateur de la courbe.

2.5.3 Problèmes liés au problème de logarithme discret

En effet, on peut aussi utiliser d'autres problèmes variants du DLP (appelés aussi les hypothèses du problème de logarithme discret dans la littérature) pour concevoir des cryptosystèmes, on cite les deux hypothèses Computational Diffie-Hellman (CDH) et Decisional Diffie-Hellman (DDH) qui sont définis comme suit :

Définition 2.5.3 (CDH)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p, g un générateur de $\mathbb{G}$. Étant donné $g, u = g^a$ et $v = g^b$ où a et b sont des aléatoires dans $\mathbb{Z}_p$ non donnés,

$$\text{CDH consiste à calculer } w = g^{ab}.$$

Définition 2.5.4 (DDH)**.** Soit $\mathbb{G}$ un groupe cyclique d'ordre premier p, g un générateur de $\mathbb{G}$. Étant donné $u = g^a$, $v = g^b$ et $w = g^c$ où a, b et c sont des aléatoires dans $\mathbb{Z}_p$ non donnés,

$$\text{DDH consiste à vérifier si } c = ab.$$

Remarque. nous avons utilisé la notation multiplicative, mais les hypothèses du problème du logarithme discret sont aussi utilisées avec le groupe de points d'une courbe elliptique.

En effet il existe d'autres hypothèses qui sont liées au couplage, on cite quelques unes dans le chapitre suivant.

2.5.4 Complexité du problème du logarithme discret

Dans un premier temps le DLP et ses variantes sont appliqués sur un groupe multiplicatif cyclique $\mathbb{Z}_p^*$ des entiers modulo un premier p mais ce groupe est problématique pour beaucoup de raisons, notamment parce que le DLP sur ce groupe n'est pas suffisamment difficile. Le meilleur algorithme connu pour résoudre le DLP appelé GNFS (pour General Number Field Sieve) s'exécute dans un temps $\exp(O((\log p)^{1/3}))$, il est utilisé en 2016 pour résoudre un DLP modulo un entier premier de 768-bit. Cet algorithme est la raison pour qu'on doive utiliser un entier premier p dont la taille est d'au mois 2048-bit, des applications de sécurité sensibles doivent utiliser des entiers premiers plus larges. L'arithmétique modulo un tel entier premier large est lente et elle diminue l'efficacité d'un cryptosystème qui l'utilise [11].

Plusieurs groupes cycliques ont été étudiés, mais le groupe de points d'une courbe elliptique s'est avéré le plus pratique et est le plus utilisé dans l'Internet aujourd'hui. Le meilleur algorithme connu pour résoudre le DLP sur le groupe de points d'une courbe elliptique d'ordre q s'exécute dans un temps $O(\sqrt{q})$. Cela signifie que lorsqu'on utilise un groupe d'ordre $q \approx 2^{256}$ on obtient un temps pour résoudre le DLP qui est égale à $\sqrt{q} = 2^{256}$, on obtient donc le même niveau de sécurité offert par le fameux cryptosystème symétrique AES-128 (présenté dans [11, 6]). l'opération de groupe qui est la multiplication scalaire utilise des opérations arithmétiques de base modulo un entier premier de 256-bit qui est plus rapide par rapport aux opérations

arithmétiques modulo 2048-bit [11].

En effet, en plus des opérations arithmétiques qui sont plus rapides, l'utilisation d'un groupe de point d'une courbe elliptique nous permet aussi de générer des paramètres et des clés plus petits, cela est intéressant pour le stockage de ces clés et paramètres et pour leur transfert dans le réseaux surtout dans des environnements restreints.

2.6 Application des courbes elliptiques dans la cryptographie

L'utilisation des courbes elliptiques a été initiée à cause de la taille des clés cryptographiques qui ne cessait d'augmenter, ainsi les nouvelles propriétés appartenant à ces courbes sont utilisées pour la procédure de l'échange des clés et les signatures numériques et d'autres cryptosystèmes.

2.6.1 Échange de clé Diffie-Hellman

L'échange de clé Diffie-Hellman noté DHKE (pour Diffie-Hellman Key Exchange) par Whitfield Diffie et Martin Hellman [2] a été proposé comme solution pour le problème de distribution des clés.

Considérons les deux participants, Alice et Bob qui veulent établir une clé secrète partagée entre eux. Pour utiliser le protocole DHKE, Alice et Bob doivent se mettre d'accord sur les deux valeurs appelées paramètres du domaine qui sont : p un nombre premier assez large et $g \in \mathbb{Z}_p^*$ un générateur.

Remarque. Les paramètres publiques du domaine peuvent être obtenu d'une tierce partie sûre et non-corrompue.

Une fois les paramètres publiés, Alice et Bob peuvent générer la clé secrète partagée comme suit :

- Alice choisit aléatoirement $a \in \mathbb{Z}_p^*$, de même Bob choisit aléatoirement $b \in \mathbb{Z}_p^*$.
- Chacun calcule sa clé publique, $A = g^a \ mod \ p$ pour Alice et $B = g^b \ mod \ p$ pour Bob.
- Ils s'échangent leurs clés publiques.
- Alice calcule la clé privé comme suit : $k_{AB} \equiv B^a$, quant à Bob $k_{AB} \equiv A^b$

Démonstration. $k_{AB} = A^b = (g^a)^b = g^{ab} = g^{ba} = (g^b)^a = B^a$ □

Le défi pour un attaquant est de calculer $K = g^{ab} \bmod p$ à partir de g, A *et* B, ce qui mène au DLP. c'est pour ça qu'on assume que p est un entier assez large assurant ainsi la complexité du calcul.

Évidement la sécurité est garantie dans le cas d'un attaquant passive, ce qui veut dire que la tierce partie ne peut qu'observer les messages transitant dans le canal et est incapable d'agir sur les message soit en les falsifiant ou en générant de nouveaux messages

2.6.2 Échange de clé Diffie-Hellman avec les courbes elliptique

Passons maintenant au DHKE avec les courbes elliptique noté ECDHKE pour (Elliptic Curve DHKE).

Soient une courbe elliptique $E/\mathbb{F}_p$ définie par ses coefficients a et b, P un générateur de la courbe définie par ses coordonnées x_P et y_P, les paramètres du domaine dans ce cas sont le nombre premier p, la courbe $E/\mathbb{F}_p$ et le générateur P.

Une fois les paramètres publiés, Alice et Bob peuvent générer la clé secrète partagée comme suit :

- Alice choisit aléatoirement $a \in \mathbb{Z}_p^*$, de même Bob choisit aléatoirement $b \in \mathbb{Z}_p^*$.
- Chacun calcule sa clé publique, $A = aP = (x_A, y_A)$ pour Alice et $B = bP = (x_B, y_B)$ pour Bob.
- Ils s'échangent leurs clés publiques.
- Alice calcule la clé privé comme suit : $k_{AB} = aB = (x_{AB}, y_{AB})$, quant à Bob $k_{AB} = bA = (x_{AB}, y_{AB})$

Démonstration. $k_{AB} = bA = b(aP) = baP = abP = a(bP) = aB$ □

Remarques.

- En général, une des coordonnées de k_{AB} est haché puis utilisé en tant que clé symétrique.
- Le choix d'une courbe elliptique n'est pas fait aléatoirement et tient compte de certaines propriétés qui garantissent la sécurité.
- L'utilisation des courbes elliptiques n'est restreinte qu'a l'échange de clé de Diffie-Hellman seulement mais peut être utilisée avec d'autre protocoles tel que la signature numérique et le chiffrement.

2.7 Conclusion

Les courbes elliptiques jouent un rôle très important dans la cryptographie de nos jours, plusieurs cryptosystèmes peuvent être réalisés à travers ces pièces mathématique magiques, ainsi, elles sont les éléments de base pour construire le couplage qu'on va présenté dans le chapitre suivant. Dans ce chapitre on pu couvrir la thématique des courbes elliptiques en donnant d'abord des rappels mathématique, puis en présentant ces courbes sur $\mathbb{R}$ et sur $\mathbb{F}_p$, on a ensuite présenté le DLP classique et en utilisant les courbes elliptiques, on a aussi présenté le ECDHKE, une des applications des courbes elliptique. Pour plus de détails mathématique sur les courbes elliptiques on recommande au lecteur de voir [12].

Chapitre 3

Couplage

Sommaire

3.1 Introduction

La première fois de son apparition, le couplage a été utilisé dans la cryptographie pour un but destructif pour attaquer le ECDLP en le réduisant en DLP où des algorithmes sous-exponentiels sont connus pour résoudre ce dernier (GNFS vue dans le chapitre précédent). Après, le couplage était utilisé pour construire de nouveaux cryptosystèmes et pour améliorer des systèmes déjà existants, le couplage est par conséquent un domaine de recherche très actif.

3.2 Définition

Soient $\mathbb{G}_1$, $\mathbb{G}_2$, et $\mathbb{G}_T$ des groupes tous du même ordre premier ($\#\mathbb{G}_1 = \#\mathbb{G}_2 = \#\mathbb{G}_T$), et soit $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$,
Un couplage est l'application e définit comme suit :

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T.$$

Qui doit garantir les propriétés suivantes :

1. **La bilinéarité :** soit $a, b \in \mathbb{Z}, e(aP, bQ) = e(P, Q)^{ab} = e(bP, aQ)$
2. **La non-dégénérescence :**
   - $\forall P\ \exists Q$ tel que $e(P, Q) \neq 1_{G_T}$,
   - $\forall Q\ \exists P$ tel que $e(P, Q) \neq 1_{G_T}$ où 1_{G_T} et l'élément neutre de groupe G_T.
3. **La calculabilité :** l'application e doit être calculée en un temps polynomial.

Remarque. Vu que $\mathbb{G}_1$ et $\mathbb{G}_2$ sont d'ordre premier alors la deuxième propriété implique que si P et G sont des générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$ respectivement alors $e(P, Q)$ est un générateur de $\mathbb{G}_T$ [13] .

Généralement, le groupe $\mathbb{G}_1$ est un sous-groupe d'un groupe additif de points d'une courbe elliptique $E(\mathbb{F}_p)$, $\mathbb{G}_2$ est un sous-groupe d'un groupe additif de points d'une courbe elliptique $E(\mathbb{F}_{p^k})$, et le groupe $\mathbb{G}_T$ est un sous-groupe de groupe multiplicatif $\mathbb{F}^*_{p^k}$, par conséquent on utilisera la notation additive pour les groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ et la notation multiplicative pour le groupe $\mathbb{G}_T$.

Un couplage peut être simplement vu comme une fonction qui prend en entrée deux points d'une courbe elliptique et retourne un élément d'un corps fini en sortie.

3.3 Types de couplage

Soit un couplage $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, selon la relation qui existe entre les groupes qui le définissent, trois types de couplage peuvent être distinguables :

Type 1. $\mathbb{G}_1 = \mathbb{G}_2$

Type 2. $\mathbb{G}_1 \neq \mathbb{G}_2$ et il existe un isomorphisme efficace connue $\phi : \mathbb{G}_2 \to \mathbb{G}_1$

Type 3. $\mathbb{G}_1 \neq \mathbb{G}_2$ et il n'existe pas un isomorphisme efficace connue entre $\mathbb{G}_1$ et $\mathbb{G}_2$

Le type 1 est appelé un couplage symétrique et est le premier à être utilisé dans les protocoles de cryptographie, les types 2 et 3 sont appelés des couplages asymétriques, on va associer dans la suite à chaque type de couplage des familles de courbes elliptiques qui sont compatibles avec.

3.4 Variantes de DLP liés au couplage

Soit un couplage symétrique $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ et P est un générateur dans $\mathbb{G}_1$. Étant donné P, aP, bP et $cP \in \mathbb{G}_1$ où a, b et c sont des aléatoires dans $\mathbb{Z}_p^*$ non donnés, on rappelle que l'hypothèse DDH vu dans le chapitre précédent consiste à vérifier que c égale à ab, Joux et Nguyen [14] ont constaté que le DDH est facile dans $\mathbb{G}_1$ en utilisant le couplage e comme suit :

$$ab = c \bmod p \iff e(P, P)^c = e(aP, bP)$$

Par conséquent le DDH ne peut pas être utilisé pour construire des cryptosystèmes, au lieu de cela une variante de l'hypothèse de CDH appelée Bilinear Diffie-Hellman (BDH) est utilisée pour construire quelques cryptosystèmes à base de couplage tel que dans [15, 13].

BDH Soit un couplage symétrique $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ et P un générateur dans $\mathbb{G}_1$. Étant donné P, aP, bP et $cP \in \mathbb{G}_1$ où a, b et c sont des aléatoires dans $\mathbb{Z}_p^*$ non donnés,

$$\text{BDH consiste à calculer } w = e(P, P)^{abc}.$$

Il existe plusieurs autres variantes de DLP liées au couplage, on ne les cite pas toutes ici mais on recommande au lecteur de voir les articles constituants la section d'état de l'art dans le chapitre suivant où chaque hypothèse utilisée est expliquée dans l'article correspondant.

3.5 Domaines d'applications de couplage

Eu plus de son effet destructif qui a permit d'attaquer la cryptographie à base des courbes elliptique dans un premier temps, la propriété de bilinéarité de couplage a permit après de concevoir des protocoles de sécurité totalement nouveaux tel que la cryptographie à base d'identité que nous lui allons consacrer le chapitre suivant, ainsi d'améliorer des protocoles déjà existants tel que l'échange de clé et la signature numérique, on présente quelques applications de couplage dans cette section.

3.5.1 Réduction de ECDLP en DLP

Menezes, Okamoto et Vanstone [16] sont les premiers à introduire le couplage dans la cryptographie en 1993 par l'attaque MOV (de leurs noms), cette dernière a pour but de résoudre le ECDLP en le convertissant en un DLP correspondant. Rappelons du chapitre 2 que le meilleur algorithme connu pour résoudre le ECDLP est de complexité exponentielle, et qu'il existe un algorithme de complexité sous-exponentielle pour résoudre le DLP classique, cette attaque affecte une famille précise de courbes elliptiques appelées les courbes elliptiques super singulières que nous allons présenter dans la suite, ces courbes sont considérées comme faibles et par conséquent il n'est pas conseillé d'utiliser une de ces courbes ni une courbe aléatoire et dans ce contexte il existe plusieurs courbes sécurisés recommandées et standardisées, on cite par exemple les courbes définies dans [17], ces courbes sont les plus utilisées à nos jours dans presque la majorité des cryptosystèmes. L'attaque MOV est décrite comme suit :
Étant donné P et xP deux points d'une courbe elliptique, ECDLP consiste à calculer x. Soit maintenant le couplage e et un autre point Q, on peut calculer $e(P, Q)$ et $e(xP, Q) = e(P, Q)^x$ (par la bilinéarité) qui sont tous les deux des éléments d'un corps fini et donc on a converti le ECDLP en DLP.

Un ans après, en 1994 Frey et Ruck [18] ont aussi attaqué le ECDLP en le convertissant en DLP de la même façon mais en utilisant un autre couplage dit de Tate qui est plus efficace.

3.5.2 Partage de clé entre trois participants en un seul tour

Nous avons vu dans chapitre 2 que l'une des applications de l'ECDLP est l'échange de clé de Diffie-Hellman, ce dernier assure que la clé soit échangée entre deux participants, et peut être étendu en un échange entre trois participants mais en utilisant deux tours où chaque participant utilise une canal pour envoyer des paramètres deux

fois pour que lui et les autres peuvent calculer la clé commune.

Joux [4] a proposé en 2000 un protocole basé sur le couplage pour l'échange de clé entre trois participants en utilisant seulement un seul tour.

Considérant les trois participants Alice, Bob et Carol, ils doivent se mettre d'accord sur $\mathbb{G}_1$ et $\mathbb{G}_T$, un couplage symétrique $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ et P un générateur de $\mathbb{G}_1$, le protocole d'échange de clés de Joux se déroule comme suit :

1. Alice choisit aléatoirement $a \in \mathbb{Z}_p^*$ et envoie aP à Bob et Carol
2. Bob choisit aléatoirement $b \in \mathbb{Z}_p^*$ et envoie bP à Alice et Carol
3. Carol choisit aléatoirement $c \in \mathbb{Z}_p^*$ et envoie cP à Alice et Bob
4. Alice, Bob et Carol calculent respectivement $e(bP, cP)^a$, $e(aP, cP)^b$ et $e(aP, bP)^c$

Remarque. Les étapes 1, 2 et 3 se déroulent en parallèle en un seul tour d'échange de messages.

La clé partagée est $e(bP, cP)^a = e(aP, cP)^b = e(aP, bP)^c = e(P, P)^{abc}$.
Un attaquant qui veut trouver la clé partagée doit résoudre un BDH.

3.5.3 Schéma de signature courte

La signature numérique est une primitive indispensable dans la cryptographie, qui sert à vérifier si un message donné est venu de la part d'une personne bien précise ou non, et donc à assurer la non répudiation. La signature numérique ne peut être utilisée que dans un environnement de cryptographie à clé publique, où une paire de clé est d'abord générée, la clé privé est utilisée pour signer un message en générant une signature, et la clé publique correspondante est utilisée pour vérifier si ce message signé vient de son vrai émetteur. On veut qu'une signature soit la plus courte possible surtout dans des environnements restreints (contraintes de réseau) mais qui doit aussi assurer un certain niveau de sécurité, réellement trois schémas de signature numérique sont utilisés, la signature RSA qui est de longueur 1024 bit, la signature DSA et la signature ECDSA qui sont toutes les deux de longueur 320 bit.

Boneh, Lynn et Shacham [15] ont proposé une signature numérique courte dite BLS (de leurs noms) en 2001, elle est de longueur 160 bit et elle assure le même niveau de sécurité qu'une signature DSA de longueur 320 bit. Une signature BLS nécessite une fonction de hachage H qui prend un message binaire d'une taille arbitraire et le transforme en un point d'une courbe elliptique comme suit $h(m) : \{0,1\}^* \to E(\mathbb{F}_p)$, il existe plusieurs fonction de hachage de ce type dans la littérature mais on ne

s'intéresse pas à leurs calculs dans ce contexte.

Soit $\mathbb{G}_1$ et $\mathbb{G}_T$, un couplage symétrique $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$, P un générateur de $\mathbb{G}_1$ et une fonction de hachage $h(m) : \{0,1\}^* \to \mathbb{G}_1$, une signature BLS pour un message de taille arbitraire m est définie par les trois algorithmes suivants :

Génération de clés

- Choisir aléatoirement $x \in \mathbb{Z}_p^*$
- La clé privée est x, et la clé publique est $Q = xP$

Signature

- On calcule d'abord le haché de m : $H = h(m)$
- La signature σ de message m est calculée comme suit : $\sigma = xH$

Vérification

- L'authentification est vraie si $e(\sigma, P) = e(H, Q)$, si non elle est fausse

Démonstration. Par la bilinéarité du couplage e on obtient : $e(\sigma, P) = e(xH, P) = e(H, P)^x = e(H, xP) = e(H, Q)$ □

Remarque. Nous rappelons qu'une l'émetteur est le seule qui peut signer son message parce qu'il possède la clé privé, et qu'une signature peut être vérifié par tous le monde.

3.6 Calcul du couplage

Jusqu'à maintenant, nous avons présenté un couplage e comme une boite noire, on va maintenant entrer dans le cœur de cette application bilinéaire.

En effet il existe deux couplages principaux connus, couplage de Weil et couplage de Tate, on va commencer d'abord par donner quelques concepts mathématiques essentiels afin de définir ces deux couplages, dans la suite on discutera des optimisations des couplages de Weil et de Tate qui sont les plus implémentées.

3.6.1 Groupe r-torsion de points

Définition 3.6.1 (r-torsion de points)**.** Soit $E(\mathbb{F}_p)$ un groupe de points d'une courbe elliptique et r un facteur premier du cardinal $E(\mathbb{F}_p)$, on appelle l'ensemble des points d'ordre r l'ensemble r-torsion de points noté $E(\mathbb{F}_p)[r]$.

$$E(\mathbb{F}_p)[r] = \{P \in E(\mathbb{F}_p) : rP = \mathcal{O}\}$$

On appelle le point P de cet ensemble un point de r-torsion.

En effet l'ensemble de points de r-torsion forme un groupe, et plus exactement $E(\mathbb{F}_p)[r]$ présente un sous-groupe de $E(\mathbb{F}_p)$.

Il peut être prouvé que pour un entier $k \geq 1$, le groupe r-torsion de points $E(\mathbb{F}_{p^k})[r]$ contient exactement r^2 [12], ce k qu'on va discuter dans la prochaine sous-section est appelé le degré de plongement, d'une autre part pour tout $k^{'} \geq k$ on a $E(\mathbb{F}_{p^{k'}})[r] = E(\mathbb{F}_{p^k})[r]$, autrement dit, il n'existe pas un autre r-torsion de points après le premier trouvé qui contient r^2 points peu importe combien de fois nous étendons le corps [19].

Pour plus de détails sur la structure de r-torsion de points ainsi que pour des démonstrations importantes on recommande au lecteur de voir le chapitre 3 de [12].

3.6.2 Degré de plongement

Le degré de plongement est un élément essentiel dans le calcul du couplage, il détermine l'ensemble où vivent les points d'un r-torsion, il est défini comme suit :

Définition 3.6.2 (Degré de plongement)**.** Soit $E(\mathbb{F}_p)$ un groupe de points d'une courbe elliptique et r un entier premier qui divise $\#E(\mathbb{F}_p)$, le degré de plongement de $E(\mathbb{F}_p)$ relativement à r est le plus petit entier positif k tel que r divise $p^k - 1$.

En effet, le degré de plongement k nous permet de déterminer le premier r-torsion qui contient r^2 points qui est exactement le groupe $E(\mathbb{F}_{p^k})[r]$ avec $\mathbb{F}_{p^k}$ l'extension de $\mathbb{F}_p$ de degré k, et on essaye de trouver des courbes qui admettent des r-torsion d'ordre r^2 avec un degré de plongement k raisonnable pour pouvoir faire le calcul du couplage efficacement dans le corps $\mathbb{F}_{p^k}$, c'est pour ça qu'on va parler après des familles bien précises de courbes qui sont adaptées au couplage.

Maintenant, on peut définir strictement les trois groupes qui construisent le couplage comme suit :

- $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$,
- $\mathbb{G}_2$ est un sous groupe de $E(\mathbb{F}_{p^k})[r]$,
- $\mathbb{G}_T = \{\mu \in \mathbb{F}_{p^k}$ tel que $\mu^r = 1\}$, autrement dit, $\mathbb{G}_T$ est le groupe des racines r-ième de l'unité.

3.6.3 Fonctions rationnelles

Les fonctions rationnelles sont des objets indispensables pour construire un couplage, on les définit comme suit :

Définition 3.6.3 (Fonction rationnelle)**.** Une fonction rationnelle f sur $E(\mathbb{F}_p)$ est une fonction qui prend un point $P \in E(\mathbb{F}_p)$ et l'associe à un élément $f(P) \in \mathbb{F}_p$:

$$f(P) : E(\mathbb{F}_p) \to \mathbb{F}_p$$

Avec $f(P)$ est écrite de la forme $f(P) = \dfrac{u(P)}{v(P)}$ où u et v sont des polynômes dans $\mathbb{F}_p[X, Y]$

Nous aurons besoin aussi de la définition suivante dans ce qui suit :

Définition 3.6.4 (Zéro de f et Pôle de f)**.** Soit $P \neq \mathcal{O}$ un point dans $E(\mathbb{F}_p)$ et f une fonction rationnelle, on appelle P zéro de f si $f(P) = 0$ (exactement si $u(P) = 0$) et on appelle P Pôle de f si f n'est pas définie en P (exactement si $v(P) = 0$).

3.6.4 Diviseurs

Les diviseurs sont aussi essentiels pour construire un couplage, on les définie comme suit :

Définition 3.6.5 (Diviseur)**.** Un diviseur D est une somme formelle de points

$$D = \sum_{P \in E(\mathbb{F}_p)} n_P(P) \text{ avec } n_P \in \mathbb{Z}^*$$

On associe à un diviseur :

- Son degré noté $deg(D)$ tel que $deg(D) = \sum_{P \in E(\mathbb{F}_p)} n_P$
- Son support noté $supp(D)$ tel que $supp(D) = \{P \in E(\mathbb{F}_p) : n_P \neq 0\}$
- Son ordre dans un point p noté $ord_P(D)$ tel que $ord_P(D) = n_P$

L'ensemble de tous les diviseurs dans $E(\mathbb{F}_q)$ noté $Div_{\mathbb{F}_q}(E)$ forme un groupe sous la loi d'addition de ces diviseurs définie comme suit :

Soit les deux diviseurs D et D' dans la courbe $E(\mathbb{F}_p)$ tel que :

$$D = \sum_{P \in E(\mathbb{F}_p)} n_P(P) \text{ et } D' = \sum_{P \in E(\mathbb{F}_p)} n'_P(P).$$

On note l'addition de D et D' par $D + D'$ tel que :

$$D + D' = \sum_{P \in E(\mathbb{F}_p)} (n_P + n'_P)(P).$$

Avec l'élément neutre le diviseur avec tout $n_P = 0$.

Définition 3.6.6 (Diviseur d'une fonction rationnelle)**.** Soit f une fonction rationnelle sur $E(\mathbb{F}_p)$, on définie son diviseur noté $Div(f)$ ou (f)

$$Div(f) = \sum_{P \in E(\mathbb{F}_p)} ord_P(f)(P).$$

Définition 3.6.7 (Diviseur principale)**.** Soit un diviseur D, s'il existe une fonction rationnelle f tel que $D = Div(f) = \sum_{P \in E(\mathbb{F}_p)} ord_P(f)(P)$ on dit que D est un diviseur principal. Un diviseur est principal si et seulement si :

- $Deg(D) = 0$
- $\sum_{P \in E(\mathbb{F}_p)} n_P(P) = \mathcal{O}$

Propriété 3.6.1. Pour tout point P appartenant à n'importe quelle extension de $E(\mathbb{F}_{p^k})$ et pour tout entier s, il existe une fonction rationnelle $f_{s,P}$ avec un diviseur :

$$(f_{s,P}) = s(P) - (sP) - (s-1)\mathcal{O}$$

En effet cette dernière propriété est utilisée dans les couplages de Weil et de Tate, on donnera plus de détails ultérieurement.

3.6.5 Couplage de Weil

Le couplage de Weil est défini comme suit :

$$\begin{aligned} e_W : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{G}_T, \\ (P, Q) &\to (-1)^r \frac{f_{r,P}(Q)}{f_{r,Q}(P)} \end{aligned}$$

3.6.6 Couplage de Tate

Le couplage de Tate est défini comme suit :

$$\begin{aligned} e_T : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{G}_T, \\ (P, Q) &\to (f_{r,P}(Q))^{\frac{p^k - 1}{r}}. \end{aligned}$$

Remarque. L'exponentiation à la puissance $\frac{p^k - 1}{r}$ est appelée l'exponentiation finale, on n'en parlera pas dans ce travail mais on recommande au lecteur le chapitre 7 de [9].

3.6.7 Algorithme de Miller

Les couplages de Weil et de Tate reposent tous les deux sur le calcul d'une fonction rationnelle de type $f_{s,P}$ qui peut être calculée par l'algorithme de Miller [20], ce dernier repose sur l'égalité de Miller qui est la suivante :

$$f_{i+j,P} = f_{i,P} \times f_{j,P} \times \frac{l_{iP,jP}}{v_{(i+j)P}}$$

où i et j sont des entiers, $l_{iP,jP}$ présente l'équation de la droite qui passe par les deux points iP et jP et $v_{(i+j)P}$ présente l'équation de la droite verticale qui passe par le point $(i+j)P$.

L'algorithme de Miller permet de calculer la fonction $f_{s,P}(Q)$ par une boucle construite sur le schéma de la multiplication par un scalaire vue dans le chapitre précédant en utilisant la méthode double-and-add, cette boucle contient $log_2(s)$ itération où dans chaque itération deux équations de droite au point Q sont effectuées, l'algorithme de Miller est présenté dans l'algorithme 3.1.

Algorithme 3.1 Algorithme de Miller

Entrée: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, et $s = (s[0], s[1], ..., s[m])_2$: entier
Sortie: $f_{s,P}(Q) \in \mathbb{G}_T$
Début
  $T \leftarrow P$;
  $f \leftarrow 1$;
  Pour i de 1 à m **Faire**
    $T \leftarrow 2T$;
    $f \leftarrow f^2 \cdot \frac{l_1(Q)}{v_1(Q)}$;
    Si $s[i] = 1$ **Alors**
      $T \leftarrow T + Q$;
      $f \leftarrow f \cdot \frac{l_2(Q)}{v_2(Q)}$;
    Fin Si
  Fin Pour
  Retourner f ;
Fin

Tel que :

- l_1 est l'équation de la tangente qui passe par T
- v_1 est l'équation de la ligne verticale qui passe par $2T$
- l_2 est l'équation de la droite (TP)
- v_2 est l'équation de la ligne verticale qui passe par $T + P$

3.7 Complexité du couplage

Les couplages de Weil et de Tate reposent tous les deux sur le calcul de l'algorithme de Miller, la complexité de ce dernier dépend de la complexité des fonctions l_1, v_1, l_2 et v_2, ces dernières dépendent du choix des paramètres tel que le degré de plongement k qui définit l'extension de corps sur laquelle se font les opérations arithmétiques et l'ordre r que l'algorithme de Miller boucle sur sa représentation binaire et qu'on veut qu'il contient le moindre de 1 possible dans sa représentation binaire (pour éviter de calculer l_2 et v_2 dans l'algorithme).

Plusieurs travaux dans la littérature qui visent à optimiser l'algorithme de Miller touchent toutes les étapes du couplage, commençant d'abord par le choix des courbes, passant par le choix de degré de plongement k et par conséquent l'extension de corps $\mathbb{F}_{p^k}$, les opérations arithmétiques sur les extension de corps...etc, l'algorithme lui même a connu des optimisations tel que l'utilisation d'une boucle construite sur le schéma de la multiplication par un scalaire Addition-Substraction au lieu du schéma double-and-add, et l'utilisation d'un nombre réduit d'itérations que nous allons voir dans la prochaine section.

En plus de l'algorithme de Miller, le couplage de Tate nécessite une exponentiation finale à la puissance $\frac{p^k - 1}{r}$, une opération coûteuse qui dépend aussi du choix de l'extension de corps choisie et des opérations arithmétiques sur cet extension de corps, et qui a aussi connu plusieurs optimisations dans la littérature.

3.8 Couplages optimaux

Nouas avons mentionné avant qu'il existe plusieurs optimisations des couplages Weil et Tate, on s'intéresse ici aux optimisations du coulage de Tate qui affectent sur la boucle de l'algorithme de Miller, on rappelle que le nombre d'itérations dans la boucle d'algorithme de Miller dans le couplage de de Tate est $\log_2(r)$, on cite ci-dessous quelques une de ces optimisations.

On aura besoin de la définition suivante :

Définition 3.8.1 (Trace de Frobenius E)**.** On appelle la valeur t la trace du Frobenius de E (trace de E) tel que :

$$t = p + 1 - \#E(F_p)$$

3.8.1 Couplage Ate

Hess, Smart et Vercauteren [21] ont introduit en 2006 une optimisation du couplage de Tate, il ont réussit à réduire le nombre d'itérations dans la boucle d'algorithme de Miller à $\log_2(T)$ tel que $T = t - 1$ où t est la trace de Frobenius. Le couplage Ate noté e_A est définit comme suit :

$$\begin{aligned} e_A : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{G}_T, \\ (P, Q) &\to (f_{T,P}(Q))^{\dfrac{p^k - 1}{r}}. \end{aligned}$$

3.8.2 Couplage Optimal-Ate

Vercauteren [22] a proposé une optimisation du couplage de Tate où il a pu prouver que la boucle la plus courte possible d'algorithme de Miller est de $\log_2(k)/\phi(12)$.

On donne l'exemple du couplage Optimal-Ate appliqué à une catégorie des courbes elliptiques adaptées au couplage appelée Barreto-Naehrig que nous allons présenté dans la section suivante et qui possède un degré de plongement $k = 12$ et est générée à partir d'un paramètre u, ce couplage noté e_{Opt} est défini comme suit :

$$\begin{aligned} e_{Opt} : \mathbb{G}_1 \times \mathbb{G}_2 &\to \mathbb{G}_T, \\ (P, Q) &\to \left(f_{6u+2,P}(Q).l_{(6u+2)Q,\pi_p(Q)}.l_{(6u+2)Q+\pi_p(Q),-\pi_p^2(Q)}\right)^{\dfrac{p^{12} - 1}{r}}, \end{aligned}$$

Tel que :

- π_p est l'application de Frobenius sur la courbe E définie comme suit : $\pi_p(x, y) \to (x^p, y^p) : E \to E$.
- $l_{Q1,Q2}$ est l'équation de la ligne correspondante à l'addition de $Q1 \in \mathbb{G}_2$ avec $Q2 \in \mathbb{G}_2$.

L'algorithme 3.2 calcule le couplage optimale appliqué sur la famille des courbes elliptiques adaptées au couplage Barreto-Naehrig, on note que cette version utilise une boucle construite sur le schéma de la multiplication par un scalaire Addition-Substraction.

Algorithme 3.2 Couplage Optimal Ate sur les courbes Barreto-Naehrig

Entrée: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$, et $s = 6u + 2 = (s[0], s[1], ..., s[m])_{NAF}$: entier
Sortie: $e_{Opt}(P, Q) \in \mathbb{G}_T$
Début
  $T \leftarrow P$; $f \leftarrow 1$;
  Pour i de 1 à m **Faire**
    $f \leftarrow f^2 \,.\, l_{T,T}(P)$; $T \leftarrow 2T$;
    Si $s[i] = -1$ **Alors**
      $f \leftarrow f \,.\, l_{T,-Q}(P)$; $T \leftarrow T - Q$;
    Si non si $s[i] = 1$ **Alors**
      $f \leftarrow f \,.\, l_{T,Q}(P)$; $T \leftarrow T + Q$;
    Fin Si
  Fin Pour
  $Q_1 \leftarrow \pi_p(Q)$;
  $Q_2 \leftarrow \pi_p^2(Q)$;
  $f \leftarrow f \,.\, l_{T,Q1}(P)$; $T \leftarrow T + Q1$;
  $f \leftarrow f \,.\, l_{T,-Q2}(P)$; $T \leftarrow T - Q2$;
  $f \leftarrow f^{(p^{12}-1)/r}$;
  Retourner f ;
Fin

3.9 Courbes adaptées au couplage

Dans la réalité, la construction des courbes elliptiques qui sont compatibles avec le couplage est une opération complexe qui dépend de plusieurs paramètres devants être soigneusement choisis pour assurer un certain niveau de sécurité d'une part et pour rendre le couplage optimisé en terme de complexité et par conséquent en terme de temps de calcul d'une autre, on essaye dans cette section de citer quelques familles des courbes compatibles avec les couplages qui sont les plus redondantes dans la littérature et les plus implémentées.

Dans ce qui suit on assumera que :

- p est la caractéristique du corps fini sur lequel la courbe est définie, r est l'ordre de la courbe et t est la trace de Frobenius de la courbe que nous allons définir tout de suite, en effet et dans la littérature, ces trois paramètres définissent les courbes elliptiques adaptées au couplage.
- Un paramètre $u \in \mathbb{Z}$ est utilisé aussi dans la littérature, il joue le rôle d'une graine pour générer p, r et t, ce u doit être choisi de tel sorte que r et p soient premiers entre eux et garantissent un certain niveau de sécurité voulu.

3.9.1 Courbes super singulières

Définition 3.9.1 (Courbe super singulière)**.** Soit Une courbe elliptique E définie sur un corps fini $\mathbb{F}_p$, E est dite super singulière si

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

Il existe bien évidemment d'autre conditions équivalentes à satisfaire.

Une courbe E est dite super singulière si sa trace du Frobenius est un multiple de la caractéristique de $\mathbb{F}_p$

3.9.2 Courbes BLS

Barreto, Lynn et Scott [23] ont introduit en 2003 une méthode pour générer des courbes elliptiques sur un corps fini $E(\mathbb{F}_p)$ adaptées au couplage avec un degré de plongement $k = 12$.
Une courbe BLS12 est définie par :

$$E : y^2 = x^3 + b$$

Les paramètres p, r et t sont obtenu par les équations suivantes :

$$\begin{aligned} p &= (u-1)^2(u^4 - u^2 + 1)/3 + u \\ r &= u^4 - u^2 + 1 \\ t &= u + 1 \end{aligned}$$

3.9.3 Courbes BN

En 2005, Barreto et Naehrig [24] ont découvert une méthode pour construire des courbes elliptiques adaptées au couplage avec un degré de plongement $k = 12$ définies par l'équation :

$$E : y^2 = x^3 + b, b \neq 0$$

Les paramètres p, r et t sont obtenu par les équations suivantes :

$$\begin{aligned} p &= 36u^4 + 36u^3 + 24u^2 + 6u + 1 \\ r &= 36u^4 + 36u^3 + 18u^2 + 6u + 1 \\ t &= 6u^2 + 1 \end{aligned}$$

3.9.4 Courbes KSS

Kachisa, Schaefer et Scott [25] ont proposer en 2008 l'utilisation des courbes adaptées au couplage avec un degré de plongement $k = \{16, 18, 32, 36, 40\}$.
La courbe KSS-16 avec un degré de plongement $k = 16$ par exemple est définie comme suit :

$$E/\mathbb{F}_{p^{16}} : Y^2 = X^3 + aX$$

Tel que $a \neq 0 \in F_p$ et $X, Y \in \mathbb{F}_{p^{16}}$.

Les paramètres p, r et t sont obtenu par les équations suivantes :

$$p = (u^{10} + 2u^9 + 5u^8 + 48u^6 + 152u^5 + 240u^4 + 625u^2 + 2398u + 3125)/980$$
$$r = (u^8 + 48u^4 + 625)/61255$$
$$t = (2u^5 + 41u + 35)/35$$

3.10 Conclusion

On a présenté dans ce chapitre des généralités sur les couplages, ces applications bilinéaires qui ont plusieurs applications dans la cryptographie moderne. On rappelle que le couplage est une notion très vaste et qu'on peut pas couvrir toutes ses composantes en détails dans ce mémoire, et donc on recommande au lecteur de voir [19, 26, 9] pour plus de détails sur les points qu'on a vu dans ce chapitre, ainsi que des points qu'on a pas vu tel que le twist d'une courbe elliptique, l'exponentiation finale utilisée dans le couplage dans de Tate et ses variantes et des optimisations qui touchent toutes les parties d'un couplage, et on rappelle aussi que des démonstrations importantes sont dans [12].

Chapitre 4

Cryptographie à base d'identité

Sommaire

4.1 Introduction

De nos jours, le système de gestion des clés publiques basé sur les certificats appelé PKI (Public Key Infrastructure) est toujours en opérabilité. Cependant, dû aux problèmes liés au stockage de ces clés et à la nécessité de très larges bases de données, Shamir [5] proposa un schéma de cryptosystèmes à base d'identité, se servant de l'identité de l'utilisateur une clé publique et de s'en passer des clés traditionnelles. Dans ce chapitre on présente ce schéma en définissant son contexte de sécurité et sa structure et en abordant quelques travaux accomplis dans ce sujet.

4.2 Chiffrement à base d'identité

Le chiffrement à base d'identité (en anglais Identity-Based Encryption : IBE) a été introduit par Shamir [5] en 1984 qui voulait une approche différente, passant par quatre algorithmes : **Setup, Extract, Encrypt, Decrypt** (respectivement : **Initialiser, Extraire, Chiffrer, Déchiffrer**), où la clé publique est une chaîne de caractères extraite de l'identité du récepteur (sa date de naissance, son adresse mail, etc.), mais la concrétisation consistante de cette idée n'a vu le jour que 17 ans après avec la proposition d'un schéma de chiffrement basé sur l'identité par Boneh et Franklin [13]. Depuis, plusieurs propositions ont suivi.

4.2.1 Structure d'un IBE

Dans un schéma de chiffrement basé sur l'identité quatre algorithmes ou protocoles sont nécessaires :

Initialiser(λ) **:** C'est l'initialisation du schéma, où à partir du paramètre de sécurité λ on obtient deux clés maîtresses publique et secrète respectivement mpk et msk pour (master public key et master secret key). Bien évidement la mpk est publique tandis que la msk est gardée secrète sauf pour le générateur de clés privées PKG(Private Key Generator).

Extraire(msk, ID) **:** Lancé par le PKG, cet algorithme prend en entrée la clé maîtresse secrète msk et un identifiant du destinataire ID $\in \{0,1\}$ et renvoie au final une clé privée sk_{ID} pour le déchiffrement.

Chiffrer(mpk, ID, m) **:** À partir de la clé publique mpk et de l'identifiant ID, le message m est chiffré, retournant ainsi un message chiffré c.

Déchiffrer(mpk, sk_{ID}, c) **:** C'est un algorithme déterministe qui à partir des paramètres mpk, sk_{ID} et du chiffré c, renvoie soit le message m soit une exception d'erreur $\perp$.

4.2.2 Définitions

Avant de continuer nous allons donner quelques définitions nécessaires pour la suite.

Fonction de hachage cryptographique :

Une fonction de hachage est un algorithme déterministe et efficace qui calcule un haché de taille fixe à partir d'un message binaire d'une taille arbitraire.

$$H(m) = h : \{0,1\}^* \rightarrow \{0,1\}^n$$

Pour être cryptographique, une fonction de hachage doit vérifier les conditions suivantes :

- **Résistante à la pré-image :** étant donné un haché h, il est impossible de trouver un message m tel que $H(m) = h$.
- **Résistante à la seconde pré-image :** étant donné un message m et un haché h tel que $H(m) = h$, il est impossible de trouver un autre message m' tel que $H(m') = H(m)$.
- **Résistante à la collision :** il est impossible de trouver deux messages m et m' tel que $m \neq m'$ et $H(m') = H(m)$.

PKI :

La PKI(Public key infrastructure) est une autorité qui en plus de la gestion des clés publiques, fournit et gère la garantie d'identité numérique aux utilisateurs, Cette identité numérique est appelée certificat numérique, elle contient la clé publique de l'utilisateur ainsi que son identité. Le certificat est construit à la demande de l'utilisateur, est sera signé avec sa clé privée.

Modèle d'oracle aléatoire :

Dans un modèle d'oracle aléatoire, un algorithme public et accessible (souvent une fonction de hachage) est considérée comme oracle aléatoire. Ce qui veut dire qu'un adversaire ne peut distinguer ce que cet algorithme donne en sortie, d'une sortie d'un algorithme purement aléatoire, même si en réalité il n'existe aucun oracle

purement aléatoire. Un tel modèle permet de garantir un niveau de sécurité raisonnable, mais on l'évite pour un haut niveau de recherche.

Modèle standard :

À l'opposé du modèle d'oracle aléatoire, ce modèle est plus convoité. Dans ce modèle la sécurité est basée seulement sur la complexité (Temps, matériel).

Sécurité forte :

Dans ce modèle de sécurité l'adversaire choisit les identités attaqués après avoir eu accès au paramètres publiques du système.

Sécurité sélective-ID :

Dans ce modèle de sécurité l'adversaire est contraint de choisir les identités attaqué avant que les paramètres publiques ne soient générés (avant même le lancement du système). Le niveau de sécurité est donc plus faible que celui du modèle de la sécurité forte (full security).

4.3 Sécurité sémantique d'un IBE

Dans un cryptosystème, la sécurité parfaite est impossible à garantir c'est pour ça qu'on parle plutôt de **sécurité sémantique**.

Sécurité sémantique :

*É*tant donné deux messages m_1, m_2 que l'adversaire soumet au « challenger », le « challenger » chiffre les deux messages et renvoie le chiffré c de l'un des deux à l'adversaire, le cryptosystème est dit sémantiquement sécurisé si et seulement si l'adversaire est dans l'incapacité de déterminer de quel message s'agit il.

$$Adv_A|Pr_A(c = Chiffrer(m_1)) - Pr_A(c = Chiffrer(m_2)| \leq \epsilon.$$

La sécurité sémantique dans un schéma de chiffrement à base d'identité ne s'éloigne pas de la définition générale, le modèle de cette sécurité est proposé la première fois par Boneh et Franklin [13]. Le jeu de sécurité varie selon l'attaque choisie par l'adversaire.

4.3.1 IND-ID-CCA

Dans un modèle de sécurité forte où l'adversaire choisit les identités attaquées après avoir obtenu les paramètres publiques. L'IND-ID-CCA (Indistinguishability Identity CCA)c'est l'indistinguabilité contre une attaque à identité choisie et texte chiffré choisi CCA(Chosen Cipher-text Attack).
Soient un « challenger » $\mathcal{C}$ et un adversaire $\mathcal{B}$.

L'IND-ID-CCA est définie par le jeu de sécurité suivant :

Setup : $\mathcal{C}$ exécute **Initialiser**(λ), ce qui permet à $\mathcal{B}$ d'obtenir la clé maîtresse publique mpk, la clé maîtresse secrète (msk) quant à elle n'est pas divulguée.

Phase 1 : $\mathcal{B}$ effectue des requêtes $q_1, q_2, ..., q_i$ où chaque requête q_j est composée de deux sous-requêtes qui sont effectuées de façon adaptative :

- requête pour une clé privée de l'identité id_j. $\mathcal{C}$ exécute l'extraction de la clé privée **Extraire**(msk, id_j) et retourne la sk_{id_j} correspondante à $\mathcal{B}$
- requête pour le déchiffrement d'un chiffré $\mathfrak{C}_{id_j}$, avec l'identité id_j.$\mathcal{C}$ exécute **Extraire**(msk, id_j), puis lance le déchiffrement **Déchiffrer**(mpk, id_j, sk_{id_j}, $\mathfrak{C}_{id_j}$) et retourne le résultat à l'adversaire.

Challenge : Quand $\mathcal{B}$ décide que la première phase est terminée, il choisit une identité id^* tel que $id^* \notin id_1, ..., id_i$, ainsi que deux message de longueur égale $m_0, m_1 \in M_\lambda$. $\mathcal{C}$ chiffre les deux messages avec **Chiffrer**(mpk, id^*, m_0), **Chiffrer**(mpk, id^*, m_0) et renvoie aléatoirement le chiffré de l'un des deux à l'adversaire $\mathfrak{C}^*$.

Phase 2 : $\mathcal{B}$ effectue d'autres requêtes $q_{i+1}, q_{i+2}, ..., q_m$ à la seul condition qu'il ne doit pas inclure id^* dans la liste des id qu'il choisira ($id_n \neq id^*$ tel que $n \in \{i+1, .., m\}$). $\mathcal{C}$ répond de la même manière qu'en première phase.

Deviner : Au final, $\mathcal{B}$ devra dire de quel message le chiffré $\mathfrak{C}^*$ provient, il gagne le challenge si sa déduction est vraie et faite dans un temps raisonnable, sinon il perd.

On dit qu'un IBE est sémantiquement sécurisé contre une attaque CCA (IND-ID-CCA secure) si pendant un temps polynomial l'avantage :

$$Adv_A|Pr_A(\mathfrak{C}^* = \textbf{Chiffrer}(mpk, id^*, m_0)) - Pr_A(\mathfrak{C}^* = \textbf{Chiffrer}(mpk, id^*, m_1))| \text{ est négligeable.}$$

4.3.2 IND-ID-CPA

La sécurité sémantique d'un IBE définie par un jeu IND-ID-CPA est identique à celle du IND-ID-CCA à l'exception que l'adversaire ne fait pas de requête de déchiffrement.
Soient un « challenger » $\mathcal{C}$ et un adversaire $\mathcal{B}$, le jeu est comme suit :

Setup : $\mathcal{C}$ exécute **Initialiser**(λ),ce qui permet à $\mathcal{B}$ d'obtenir la clé maîtresse publique mpk, la clé maîtresse secrète (msk) quant à elle n'est pas divulgué.

Phase 1 : $\mathcal{B}$ effectue des requêtes $q_1, q_2, ..., q_i$ de façon adaptative où chaque requête q_j est un requête pour obtenir une clé privée de l'identité id_j. $\mathcal{C}$ exécute l'extraction de la clé privée **Extraire**(msk, id_j) et retourne la sk_{id_j} correspondante à $\mathcal{B}$.

Challenge : Quand $\mathcal{B}$ décide que la première phase est terminée, il choisit une identité id^* tel que $id^* \notin id_1, ..., id_i$, ainsi que deux message de longueur égale $m_0, m_1 \in M_\lambda$. $\mathcal{C}$ chiffre les deux messages avec **Chiffrer**(mpk, id^*, m_0),**Chiffrer**(mpk, id^*, m_0)et renvoie aléatoirement le chiffré de l'un des deux à l'adversaire $\mathfrak{C}^*$.

Phase 2 : $\mathcal{B}$ effectue d'autres requêtes $q_{i+1}, q_{i+2}, ..., q_m$ à la seul condition qu'il ne doit pas inclure id^* dans la liste des id qu'il choisira($id_n \neq id^*$ tel que $n \in \{i+1, ..., m\}$). $\mathcal{C}$ répond de la même manière qu'en première phase.

Deviner : Au final, $\mathcal{B}$ devra dire de quel message le chiffré $\mathfrak{C}^*$ provient, il gagne le challenge si sa déduction est vraie et faite dans un temps raisonnable, sinon il perd.

On dit qu'un IBE est sémantiquement sécurisé contre une attaque CCA (IND-ID-CPA secure) si pendant un temps polynomial l'avantage :

$$Adv_A|Pr_A(\mathfrak{C}^* = \textbf{Chiffrer}(mpk, id^*, m_0)) - Pr_A(\mathfrak{C}^* = \textbf{Chiffrer}(mpk, id^*, m_1))| \text{ est négligeable.}$$

4.4 État de l'art

4.4.1 Schéma de Boneh et Franklin

Principe

Boneh et Franklin [13] ont proposé en 2001 un schéma IBE qui repose sur le couplage de Weil et se sert du problème BDH pour assurer son sécurité.

Fonctionnement

Voici comment cet IBE fonctionne :

Initialiser : étape 1 : On choisit un premier p de k-bits tel que $p = 2 \bmod 3$ et $p = 6q - 1$ pour un entier premier $q > 3$.

Soit E la courbe elliptique définie par $y^2 = x^3 + 1$ sur $\mathbb{F}_p$. Choisir arbitrairement un P $\in E/\mathbb{F}_p$ d'ordre q.

étape 2 : tirer un aléa $s \in \mathbb{Z}_q^*$ et calculer $P_{pub} = sP$

étape 3 : Choisir quatre fonctions de hachage H, H_1, G et G_1 tel que :

$H : \{0,1\}^n \to \mathbb{F}_{p^2}$ où n est la taille du message m.

$H_1 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{F}_q$

$G : \{0,1\}^* \to \mathbb{F}_p$

$G_1 : \{0,1\}^n \to \{0,1\}^n$

Les paramètres publiques sont donc $mpk := (p, n, P, P_{pub}, H, H_1, G, G_1)$ et la clé maîtresse secrète est $msk := s$.

Extraire : Pour obtenir la clé secrète du déchiffrement sk_{ID} on procède comme suit :

étape 1 : convertir l'ID de chaîne de caractères en un point :

1. Calculer $y_0 = G(ID)$ et $x_0 = (y_0^2 - 1)^{1/3} = (y_0^2 - 1)^{(2p-1)/3}) \bmod p$
2. Soit $Q = (x_0, y_0)$. Calculer $Q_{ID} = 6Q$ tel que Q_{ID} est d'ordre q.

Cet algorithme de conversion est appelé $MapToPoint_G$

étape 2 : Calculer la $sk_{ID} = sQ_{ID}$, où s est la msk.

Chiffrer : Afin de chiffrer un message m de taille n destiné à l'utilisateur avec l'identité ID, On choisit un aléa $\sigma \in \{0,1\}^n$ et calculer $r = H_1(\sigma, m)$ pour obtenir un chiffré en trois parties :

$$c := (c_1, c_2, c_3) = (rP, \sigma \oplus H(g_{ID}^r), m \oplus G_1(\sigma))$$

Où $g_{ID} = e(Q_{ID}, P_{pub}) \in \mathbb{F}_{p^2}$

Déchiffrer : Pour obtenir le message original m à partir du chiffré c. On vérifie tout d'abord si c_1 appartenant à $E/\mathbb{F}_p$ est d'ordre q, si ce n'est pas le cas le chiffré est rejeté.

Dans le cas contraire le déchiffrement se fait comme suit :

1. Calculer $\sigma = c_2 \oplus H(e(sk_{ID}, c))$.
2. Calculer $m = c_3 \oplus G_1(\sigma)$.
3. Mettre $r = H_1(\sigma, m)$ et tester si $rP = c_1$ sinon rejeter le chiffré.
4. Renvoyer le message en claire m.

4.4.2 Schéma de Boneh et Boyen

Principe

Conçu par Boneh et Boyen [27] en 2004, cet IBE s'inspire d'un autre IBE par les mêmes auteurs de celui-ci, et y ajoute deux autres propriétés : (1) une réduction étroite de la sécurité sans oracles aléatoires et (2) et une indifférence à l'ordre hiérarchique. Cet IBE utilise un système de sécurité qui ne dépends pas des oracles aléatoires, il est basé sur le problème BDH et utilise une fonction de hachage résistante à la collision.

Fonctionnement

Soient :

- Un groupe $\mathbb{G}$ d'ordre p.
- Les messages chiffrés sont des éléments du groupe $\mathbb{G}_1$
- Une application de couplage e tel que $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$.
- Un ensemble $\Sigma = \{1, ..., s\}$ d'alphabet de taille s. Une famille de fonctions de hachage $H_k : \{0,1\}^w \to \Sigma^n$.

Supposant que les ID sont des éléments appartenant à $\{0,1\}^w$.

L'IBE fonctionne selon l'algorithme suivant :

Initialiser $(\mathbb{G}, \mathbb{G}_1, e)$: Pour générer les paramètres du système :

1. L'algorithme tire un générateur aléatoire $g \in \mathbb{G}$, un élément aléatoire $\alpha \in \mathbb{Z}_p$
2. Met $g_1 = g^\alpha$.
3. Tire un autre élément aléatoire $g_2 \in \mathbb{G}$ et construit une matrice U de taille $n \times s$ tel que $U = (u_i, j) \in \mathbb{G}^{n \times s}$.
4. Choisit une clé aléatoire $k \in \mathcal{K}$ pour la fonction de hachage.

Les paramètres publiques sont donc $mpk := (g, g_1, g_2, U, k)$ et la clé maîtresse secrète est $msk := g_2^\alpha$

Extraire (mpk, ID, msk) : Pour générer la clé secrète de déchiffrement pour l'identité ID $\in \{0,1\}^w$:

1. L'algorithme met $\vec{a} = H_k(ID) = a_1...an \in \Sigma^n$.
2. Tire ensuite un aléa $r_1, ..., r_n \in \mathbb{Z}_p$.

La clé secrète de déchiffrement sk_{ID} est :

$$sk_{ID} = \left(g_2^{\alpha} \cdot \Pi_{i=1}^{n} u_{i,a_i}^{r_i}, g^{r_1}, ..., g^{r_n}\right) \in \mathbb{G}^{n+1}$$

Chiffrer (mpk, ID, m) : Pour chiffrer un message $m \in \mathbb{G}_1$ destiné à l'utilisateur avec l'identité ID $\in \{0,1\}^w$,

1. L'algorithme met $\vec{a} = H_k(ID) = a_1...an \in \Sigma^n$.
2. Tire ensuite un aléa $t \in \mathbb{Z}_p$ et retourne le chiffré :

$$c = (e(g_1, g_2)^t \cdot m, g^t, u_{1,a_1}^t, g_1, g_2) \in \mathbb{G}^1 \times \mathbb{G}^{n+1}$$

Pour alléger les calculs durant le chiffrement, le résultat du couplage $e(g_1, g_2)$ peut être pré-calculé à l'avance ou mis dans l'ensemble des paramètres de la mpk.

Déchiffrer (mpk, sk_{ID}, c) : Permet d'obtenir le message en claire d'un chiffré $c = (A, B, C_1, ..., C_n)$ en utilisant la clé secrète de déchiffrement $sk_{ID} = (sk_0, sk_1, ..., sk_n)$, l'algorithme calcule et renvoie :

$$m = A \cdot \frac{\Pi_{j=1}^{n} e(C_j, sk_j)}{e(B, sk_0)}$$

4.4.3 Schéma de Waters

Principe

Proposé par Brent Waters [28] en 2009, ce système repose sur la notion de « Dual system encryption » en exploitant des problèmes similaires au problème du logarithme discret qui sont le Bilinear Diffie-Hellman et le Decisional Linear.

Fonctionnement

Cet IBE fonctionne comme suit :

Initialiser : Soit $\mathbb{G}$ un groupe d'ordre premier p.

L'algorithme choisit des générateurs $g, v, v_1, v_2, w, u, h \in \mathbb{G}$ et des exposants $a_1, a_2, b, \alpha \in \mathbb{Z}_p$. On met $\tau_1 = vv_1^{a_1}, \tau_2 = vv_2^{a_2}$.

Les paramètres publiques sont donc :

$$mpk := (g^b, g^{a_1}, g^{a_2}, g^{b \cdot a_1}, g^{b \cdot a_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, u, h, e(g,g)^{\alpha \cdot a_1 \cdot b}).$$

Et la clé maîtresse secrète est $msk := (g, g^\alpha, g^{\alpha \cdot a_1}, v, v_1, v_2)$.

Extraire (msk, $\mathcal{I}$) : L'algorithme choisit des aléas $r_1, r_2, z_1, z_2, tag_k \in \mathbb{Z}_p$. Met $r = r_1 + r_2$ et crée ensuite : $D_1 = g^{a_1 \cdot \alpha} v^r$,
$D_2 = g^{-\alpha} v_1^r g^{z_2}$,
$D_3 = (g^b)^{z_1}$,
$D_4 = v_2^r g^{z_2}$,
$D_5 = (g^b)^{z_2}$,
$D_6 = g^{r_2 \cdot b}$,
$D_7 = g^{r_1}$,
$K = (u^{\mathcal{I}} w^{tag_k} h)^{r_1}$.
La clé secrète de déchiffrement est $sk = D_1, ..., D_7, K, tag_k$

Chiffrer (mpk, $\mathcal{I}$, m) : Pour chiffrer un message $m \in \mathbb{G}$:

1. L'algorithme tire quatre aléas s_1, s_2, t et tag_c appartenant à $\mathbb{Z}_p$.
2. Met $s = s_1 + s_2$.
3. « Déguise » (blind) le message m tel que $c_0 = m \cdot (e(g,g)^{\alpha \cdot a_1 \cdot b})^{s_2}$ et crée :

$C_1 = (g^b)^{s_1 + s_2}$,
$C_2 = (g^{b \cdot a_1})^{s_1}$,
$C_3 = (g^{a_1})^{s_1}$,
$C_4 = (g^{b \cdot a_2})^{s_2}$,
$C_5 = (g^{a_2})^{s_2}$,
$C_6 = \tau_1^{s_1} \tau_2^{s_2}$,
$C_7 = (\tau_1^b)^{s_1} (\tau_2^b)^{s_2} w^{-t}$,
$E_1 = (u^{\mathcal{I}} w^{tag_k} h)^t$, $E_2 = g^t$.

Le chiffré est donc $CT = C_0, ..., C_7, E_1, E_2, tag_c$

Déchiffrer (CT,$K_{\mathcal{I}}$) : L'algorithme est capable de déchiffrer le chiffré CT pour l'identité $\mathcal{I}$ si le tag_c du chiffré est différent du tag_k de la clé de déchiffrement. Puisque les deux tags sont tirés de manière aléatoire il y a une très faible probabilité qu'ils soient identiques.

L'algorithme se décompose en plusieurs calculs. Il calcule en premier lieu :

$$A_1 = e(C_1, D_1) \cdot e(C_2, D_2) \cdot e(C_3, D_3) \cdot e(C_4, D_4) \cdot e(C_5, D_5)$$
$$= e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(v,g)^{b(s_1+s_2)r} e(v_1,g)^{a_1 b s_1 r} e(v_2,g)^{-r_1 t}.$$

Rappelons que $r = r_1 + r_2$. Ensuite il calcule

$$A_1 = e(C_6, D_6) \cdot e(C_7, D_7)$$
$$= e(v,g)^{b(s_1+s_2)r} e(v_1,g)^{a_1 b s_1 r} e(v_2,g)^{a_2 b s_2 r} \cdot e(g,w)^{-r_1 \cdot t}.$$

En mettant $A_3 = A_1/A_2 = e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(g,w)^{r_1 \cdot t}$, et si $tag_c \neq tag_k$ alors l'algorithme peut calculer :

$$A_4 = (e(E_1, D_7)/e(E_2, K))^{1/(tag_c 0 - tag_k)} = e(g,w)^{r_1 \cdot t}.$$

On peut finalement obtenir m le message en claire du chiffré CT en calculant :

$$m = C_0/(A_3/A_4)$$

4.4.4 Schéma de Wee

Principe

Wee [29] a proposé en 2015 un IBE qui fonctionne avec des groupes bilinéaire d'ordre composé et nécessitant qu'une seule application de couplage lors du déchiffrement, ce schéma est une amélioration de celui de Lewko et Waters [30].

Fonctionnement

Cet IBE fonctionne comme suit :

Initialiser ($\mathbb{G}$) : Soit $\mathbb{G} := (N, G, H, G_T, e)$, Oú N est le composé du produit de 2 nombres premiers distincts ($N = p_1 p_2$), G, H et G_T sont des groupes cycliques d'ordre N, et $e : G \times H \to G_T$ est une application de couplage bilinéaire non-dégénéré.

Aussi G_{p_1}, G_{p_2} des sous-groupes de G contenant les premiers respectivement p_1, p_2 et H_{p1} un sous-groupe de H.
On tire 3 aléas α, g_1 et u, tel que $\alpha \in \mathbb{Z}_N$ et les deux générateurs $g_1 \in G_{p_1}$ et $u \in \mathrm{H}_{p_1}$.

Soit $\mathcal{H}$ une fonction de hachage tel que $\mathcal{H} : G_T \to \{0,1\}^\lambda$.
Les paramètres publiques sont donc $mpk := (g_1, g_1^\alpha, e(g_1, u))$.
Et la clé maîtresse secrète $msk := (\alpha, u)$.

Extraire (msk, ID) : Met la clé secrète de déchiffrement $sk_{ID} = u^{\frac{1}{\alpha+ID}}$.

Chiffrer (mpk, id) L'algorithme tire aléatoirement un nombre $s \in \mathbb{Z}_N$ et renvoie le chiffré (CT, $\mathcal{K}$) = $(g_1^{(\alpha+ID)s}, \mathcal{H}(e(g_1, u)^s))$.

Déchiffrer (sk_{ID}, CT) : L'algorithme calcule et renvoie $\mathcal{H}(e(CT, sk_{ID}))$

4.4.5 Schéma de Karatsuma et Yamada

Principe

Karatsuma et Yamada [31] ont proposé un schéma IBE en 2016 réalisé en utilisant le couplage, la sécurité de leur schéma est assurée par le problème 3-CBDHE (3-Computational Bilinear Diffie-Hellman Exponent), cela est le premier IBE sécurisé sous un problème de calculabilité (ou dit de recherche) plutôt qu'un problème de décision.

Fonctionnement

Le schéma de Karatsuma et Yamada en utilisant le couplage est décrit comme suit :

Initialiser (1^λ) : En entrée 1^λ, on choisit les groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ d'ordre premier $p = p(\lambda)$, un couplage $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ et g et h générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$, respectivement, on tire $w_0, w_{1,1}, ..., w_{1,l}, w_{2,1}, ..., w_{2,l}, \alpha, \beta \leftarrow \mathbb{Z}_p$ et $rand \leftarrow \{0,1\}^{|\mathbb{G}_T|}$, la sortie est comme suit :

$mpk = (g, W_0 = g^{w_0}, \{W_{1,i} = g^{w_{1,i}}\}_{i=1}^l, \{W_{2,i} = g^{w_{2,i}}\}_{i=1}^l, g^\alpha, h^\beta, rand)$ et
$msk = (h, \alpha, \beta, w_0, w_{1,1}, ..., w_{1,l}, w_{2,1}, ..., w_{2,l})$.

Dans la suite on utilise la fonction déterministe $H : ID \to \mathbb{Z}_p$ qui est définie comme suit :

$$H(ID) = w_0 + \sum_{(i,j) \in S(ID)} w_{1,i} w_{2,i} \in \mathbb{Z}_p$$

Extraire (mpk, msk, ID) : On calcule d'abord $H(ID)$ en utilisant msk et on tire aléatoirement $r \leftarrow \mathbb{Z}_p$, on retourne alors :

$$sk_{ID} = (A_1 = h^{\alpha\beta + r.H(ID)}, A_2 = h^{-r}, \{B_j = h^{rw_{2,j}}\}_{j=1}^{l}).$$

Chiffrer (mpk, ID, m) : Afin de chiffrer un message $m \in \{0,1\}$. On tire aléatoirement $s, t_1, ..., t_l \leftarrow \mathbb{Z}_p$, et on calcule :

$C_0 = m \oplus GL(e(g^\alpha, h^\beta), rand)$,
$C_1 = g^s, C_2 = W_0^s . \prod_{j \in [1,l]} W_{2,j}^{tj}$,
$D_j = g^{tj} . \left(\prod_{i \in \{i \in [1,l] | (i,j) \in S(ID)\}} W_{1,i} \right)^{-s}$ pour $j \in [1, l]$.

On retourne à la fin $C = (C_0, C_1, C_2, \{D_{j=1}^l\})$.

Déchiffrement (mpk, sk_{ID}, C) : Afin de déchiffrer un chiffré $C = (C_0, C_1, C_2, \{D_{j=1}^l\})$ en utilisant une clé privée $sk_{ID} = (A_1, A_2, \{B_j\}_{j=1}^l)$ on calcule :

$$e(C_1, A_1).e(C_2, A_2). \prod_{j \in [1,l]} e(D_j, B_j) = e(g, h)^{s\alpha\beta}.$$

On peut maintenant récupérer le message $m = C_0 \oplus GL(e(g, h)^{s\alpha\beta}, rand)$.

4.4.6 Schéma de Watanabe, Emura et Seo

Principe

Watanabe, Emura et Seo [32] ont proposé un schéma IBE à base de couplage qui utilise des paramètres publiques d'une taille constante, leur schéma IBE est une version modifiée du schéma de Jutla-Roy [33] et est sécurisé sous les hypothèses ADDH1 et DDH2.

Fonctionnement

Le fonctionnement de cet IBE est comme suit :

Initialiser (λ) : On choisit un premier p, les groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ d'ordre premier, un couplage $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ et g_1 et g_2 générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$,respectivement

1. On tire $x_0, y_0, x_1, y_1, x_2, y_2, x_3, y_3 \leftarrow \mathbb{Z}_p$ et $\alpha, \beta \leftarrow \mathbb{Z}_p^\times$.
2. On calcule :

$$z = e(g_1, g_2)^{-x_0\alpha+y_0}, u_1 = g_1^{-x_1\alpha+y_1}, w_1 = g_1^{-x_2\alpha+y_2}, h_1 = g_1^{-x_3\alpha+y_3}, \chi_1 = g_1^{\beta(-x_0\alpha+y_0)}.$$

3. On retourne :

$mpk = (g_1, g_1^{\alpha}, u_1, w_1, h_1, \chi_1, g_2, g_2^{x_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_1}, g_2^{y_2}, g_2^{y_3}, z, g_2^{x_0\beta}, g_2^{y_0\beta}, g_2^{\frac{1}{\beta}})$,
$msk = (g_2^{y_0}, g_2^{-x_0})$.

Extraire (mpk, msk, ID) : En entrée $msk = (d_1^{'}, d_2^{'})$.

On choisit $r \leftarrow \mathbb{Z}_p$ et on calcule :

$D_1 = (g_2^{y_2})^r, D_1^{'} = d_1^{'}((g_2^{y_1})^{ID} g_2^{y_3})^r$
$D_2 = (g_2^{x_2})^{-r}, D_2^{'} = d_2^{'}((g_2^{x_1})^{ID} g_2^{x_3})^{-r}, D_3 = g_2^r.$

On retourne $sk = (D_1, D_1^{'}, D_2, D_2^{'}, D_3)$.

Chiffrer (mpk, ID_s, m) : On choisit $t, tag \leftarrow \mathbb{Z}_p$, pour $m \in \mathbb{G}_T$ on calcule :

$C_0 = mz^t$,
$C_1 = g_1^t$,
$C_2 = (g_1^{\alpha})^t$,
$C_3 = (u_1^{ID} w_1^{tag} h_1)^t$.

On retourne $C = (C_0, C_1, C_2, C_3, tag)$.

Déchiffrer (msk, sk_{ID}, C) : En recevant $sk = (D_1, D_1^{'}, D_2, D_2^{'}, D_3)$ et $C = (C_0, C_1, C_2, C_3, tag)$, on calcule :

$$m = \frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{tag} D_1^{'}) e(C_2, D_2^{tag} D_2^{'})}$$

4.4.7 Schéma de Wang et Zhao

Principe

Wang et Zhao [34] ont proposé le premier schéma Higncryption à base d'identité (IBHigncryption), leur algorithme ne requiert pas un calcul du mpk habituel et son implémentation est possible avec un des trois types de couplage, la sécurité de leur schéma est assurée par le problème Gap-SBDH.

Fonctionnement

Le schéma Wang et Zhao utilisant le troisième type de couplage, fonctionne comme suit :

Initialiser : En entrée 1^λ.

1. On choisit les groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ d'ordre premier q, un couplage $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ et $g1$ et $g2$ générateurs de $\mathbb{G}_1$ et $\mathbb{G}_2$,respectivement.
2. On sélectionne deux fonctions de hachage cryptographiques $h1 : \{0,1\}^* \to \mathbb{G}_1$ et $h2 : \{0,1\}^* \to \mathbb{G}_2$.
3. On tire une clé maîtresse secrète $s \leftarrow \mathbb{Z}_p^*$.

la sortie est comme suit :
$msk = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g1, g2, h1, h2)$ et $msk = s$.
Le PKG met mpk publique pour tous les utilisateurs dans le système et garde msk secrète.

Extraire : En entrée les paramètres publiques du système mpk et l'identité d'un utilisateur $ID \in \{0,1\}^*$, le PKG calcule $sk = (sk_1, sk_2) = (h_1(ID)^s, h_2(ID)^s)$ et retourne ce sk comme la clé secrète associée à l'identité ID.

Chiffrer $(mpk, sk_s, ID_s, ID_r, H, m)$: Soit $SE = (K_{se}, Enc, Dec)$ un chiffrement authentifié avec des données associées (AEAD pour Authenticated Encryption With Associated Data) tel que décrit dans [2019], $m \in \{0,1\}^*$ le message à chiffrer avec les données associées $H \in \{0,1\}^*$,
et $KDF : \mathbb{G}_t \times \{0,1\}^* \to \{0,1\}^*$ une fonction de dérivation de clé où κ est l'espace de clés de K_{se}. Pour la simplicité on note par ID_s l'identité publique d'émetteur (sender) qui a la clé privée $sk_s = (sk_{s_1}, sk_{s_2}) = (h_1(ID_s)^s, h_2(ID_s)^s)$ et par ID_r l'identité publique de l'utilisateur auquel le message est destiné (receiver) qui a la clé privée $sk_r = (sk_{r_1}, sk_{r_2}) = (h_1(ID_r)^s, h_2(ID_r)^s)$.
Pour chiffrer un message m avec identité d'émetteur cachée, l'émetteur :

1. Tire $x \leftarrow \mathbb{Z}_p^*$ et calcule $X = (h_1(ID_s)^x \in \mathbb{G}_1$,
2. Calcule le secret pré-partagé $PS = e(sk_{s_1}, h_2(ID_r)^x \in \mathbb{G}_t$,
3. Dérive la clé $K_1 = KDF(PS, X \parallel ID_r) \in \kappa$,
4. Calcule $C_{AE} \leftarrow Enc_{k1}(H, ID_s \parallel m \parallel x)$,

5. Renvoie finalement $C = (H, X, C_{AE})$.

Déchiffrer (mpk, sk_s, ID_r, C) : En recevant $C = (H, X, C_{AE})$ cet algorithme :

1. calcule le secret pré-partagé $PS = e(X, sk_{r_2}) \in \mathbb{G}_t$
2. Dérive la clé $K_1 = KDF(PS, X \parallel ID_r) \in \kappa$,
3. Exécute $Dec_{k1}(H, C_{AE})$.

Si $Dec_{k1}(H, C_{AE})$ retourne $\perp$ il refuse le message, sinon il obtient $\{ID_s, m, x\}$ et accepte (ID_s, m) si $X = (h_1(ID_s)^x$, sinon il rejette le message.

4.5 Synthèse et comparaison

L'IBE de Boneh et Franklin [13] à été établi après que plusieurs propositions de schéma pour le chiffrement à base d'identité n'ont pas pu être pleinement satisfaisante soit en matière de temps ou de matériel etc.
La sécurité de cet IBE à été démontré dans l'article, où une attaque sur ce schéma induit la résolution du problème de Weil Diffie-Hellman une variante du problème du bilinear Diffie-Hellman combiné avec la notion de couplage de Weil. Cependant ce schéma ne garantit pas une sécurité fiable dans un modèle standard.

Boneh et Boyen [27] ont construit un schéma de chiffrement à base d'identité sécurisé sans oracle aléatoire et l'ont améliorer par la suite. Cet IBE repose totalement sur la difficulté de calcul d'une variante du problème BDH. Ce système a été démontré sécurisé mais seulement sous un modèle de sécurité sélective-ID.

Waters [28] dans son travail a présenté une nouvelle méthode de preuve de sécurité à l'aide d'algorithme semi-fonctionnel pour le système de chiffrement appelé « *Dual System Encryption* ». Se basant seulement sur le problème Decisional Bilinear Diffie-Hellman et Decisional Linear, se système n'utilise pas d'oracle aléatoire rendant ainsi sa complexité relative qu'aux problèmes cité au-dessus.

Se basant sur le travail de Lewko et Waters [30], Wee [29] à présenté un schéma de chiffrement à base d'identité qui repose sur la notion des groupes bilinéaires d'ordre composé, utilisant moins de paramètres (que sa soit pour les clés maîtresses ou bien l'ensemble du chiffré), garantissant une sécurité avec un modèle de sécurité forte, ainsi que l'anonymat des exposant au niveau des chiffrés et des clés secrètes, problème rencontré dans les IBE précédent reposant sur le même principe. Le document présente aussi un schéma de signature intéressant(Nous invitons le lecteur

à le consulter pour plus de détail).

Karatsuma et Yamada [31] ont réussit à introduire le premier schéma IBE qui repose sur un problème de calculabilité/recherche 3-CBDH et qui utilise en même temps des paramètres publiques d'une taille sous-linéaire par rapport au paramètre de sécurité λ, ils ont comparé leur schéma avec d'autres schémas similaires qui reposent sur des problèmes de calculabilité où il marque des résultats théoriques importants en terme d'efficacité, il offre une clé maîtresse publique, un texte chiffré et une clé privée plus courte et cela est du à l'aide du problème utilisé qui est considéré comme plus dur que les problèmes utilisés dans les autres schémas, mais ce type de schéma reste toujours plus coûteux et lourd et nécessite le calcul de plusieurs couplages.

Watanabe, Emura et Seo [32] ont proposé un schéma RIBE (pour Revocable IBE) tel que décrit dans leur papier, comme base pour cet RIBE ils ont proposé une version améliorée de l'IBE de Jutla-Roy [33], leur schéma IBE marque des avantages importants du coté de la compatibilité et d'habilitée de transformation en RIBE qui apparaît assez intéressant (on recommande au lecteur de voir leur papier), il utilise aussi des paramètres publiques constants courtes. On rappelle que leur schéma repose sur l'hypothèse SXDH qui est une combinaison des deux hypothèses ADDH1 et DDH2.

Dans leurs travail, Wang et Zhao [34] ont introduit une nouvelle primitive qu'ils ont nommé IBHigncryption et qui peut être vue comme une extension d'IBE où elle offre d'autres services plus que le chiffrement, on cite la signature numérique et la non divulgation de l'identité des participants, pour cette nouvelle primitive, ils ont présenté un modèle formel de sécurité avec lequel ils ont prouvé la sécurité de leur schéma, ce dernier est assez simple, il peut être implémenté en utilisant un des trois types de couplage et il repose sur l'hypothèse Gap-SBDH, ainsi, leur algorithme d'initialisation ne requiert pas le calcul du clé maîtresse publique habituelle. Leurs schéma est comparable avec le schéma IBE de Boneh et Franklin [13] en terme d'efficacité et est plus efficace que le schéma standardisé dans IEEE P1363.3 [35]. Il donnent aussi quelques applications possibles de leur schéma tel que le protocole de communication cellulaire 5G par exemple.

Le tableau 4.1 présente une comparaison entres les différentes IBE étudiés.

Schéma	Basé sur	Modèle	Type de Sécurité
Boneh et Franklin 2001	BDH	Oracle aléatoire	IND-ID-CPA
Boneh et Boyen 2004	BDH	Standard	IND-sID-CPA
Waters 2004	BDH et Linear	Standard	IND-ID-CPA
Wee 2015	SXDH	Oracle aléatoire	IND-ID-CPA
Karatsuma et Yamada 2016	ADDH1 et DDH2	Standard	IND-ID-CPA
Wtanabe,Emura et Seo 2017	SXDH	Standard	IND-ID-CPA
Wang et Zhao 2019	Gap-SBDH	Oracle aléatoire	IND-ID-CPA

Table 4.1 – Comparaison entre les différents IBE étudiés

Dans un système IBE, en plus de la facilité d'obtenir des clés publiques à partir des identités au lieu de stocker les clés ce qui nécessite une large base de données dans le cas d'un système PKI, il n'y a pas besoin d'émettre de requête pour des certificats à chaque tentative d'établissement de connexion. Mais l'inconvénient majeur d'un schéma IBE par rapport à un système PKI est la complexité du calcul relative au couplage. Ainsi que la nécessité d'un canal sécurisé pour le transfert des clés privées.

4.6 Conclusion

La proposition d'un tel système a été engendré par la nécessité de faciliter - alléger- la gestion des clés publiques notamment par rapport à la taille de ces dernières ou bien en matière d'authentification et d'adaptation etc.. Après la première concrétisation d'un IBE en 2001 réalisé par Boneh et Franklin considéré comme étant la réponse au défi lancé par Shamir en 1984, d'autres systèmes ont été construits se basant sur des IBE existants, ou bien créant de nouveaux concepts inexistants auparavant.

Nous avons présenté dans ce chapitre quelques schémas fondateurs et d'autres récents qui illustrent l'efficacité de ce genre de système, où chaque IBE démontre sa sécurité à un certain niveau sous des modèles et principes différents.

Chapitre 5

Contribution

Sommaire

5.1 Introduction

Après avoir présenter le couplage dans le chapitre 3 et quelques travaux concernant l'IBE dans le chapitre 4, nous présentons dans ce chapitre une approche d'un schéma IBE en utilisant le couplage avec des groupes bilinéaires d'ordre premier, nous présentons ainsi notre implémentation, nous pouvons diviser cette dernière en deux parties, la première est la partie d'implémentation du couplage et la deuxième est la partie d'implémentation du schéma IBE, avant de les présenter on parle des implémentations déjà existantes du couplage ainsi que le langage de programmation que nous avons choisi : Python.

5.2 Approche proposée

Principe

Notre approche repose sur la structure « Inversion de l'exposant » présenté la première fois par Boneh et Boyen [36], cette structure a un vaste impacte sur la conception des cryptosystèmes modernes tel que les schémas IBE. En effet, notre construction se base sur celle de Wee [29], nous avons travaillé avec un couplage asymétrique où nous avons utilisé des groupes bilinéaires d'ordre premier au lieu des groupes bilinéaires d'ordre composé tel que Wee a fait et nous avons en plus intégré un chiffrement authentifié sécurisé tel que Wang et Zhao [34] ont fait dans leur travail. Notre construction permet une sécurité adaptative sous l'hypothèse SXDH.

Fonctionnement

Nous présentons notre approche décrite comme suit :

Initialiser(λ) : En entrée λ, le PKG choisit :

- des groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_T$ d'ordre premier p en fonction de λ,
- un couplage asymétrique $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$,
- un générateur g de $\mathbb{G}_1$,
- une fonction de hachage cryptographique $H_1 : \mathbb{G}_T \to \{0,1\}^n$,
- et une fonction de hachage cryptographique $H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$.

Ensuite, le PKG tire aléatoirement :

- une clé master secrète $\alpha \leftarrow \mathbb{Z}_p^*$
- un élément $h \leftarrow \mathbb{G}_2$

La sortie est comme suit :

$$msk = (\alpha, h) \text{ et } mpk = (g, g^\alpha, e(g,h), H).$$

Extraire(msk, id) **:** Le PKG génère la clé secrète associée à id comme suit :

$$sk_{id} = h^{\frac{1}{\alpha + H_2(id)}}$$

Chiffrer(mpk, id, M) **:** Soit $SE = (K, Enc, Dec)$ un chiffrement authentifié sécurisé (AE pour Authenticated Encryption) tel que décrit dans [11] .

L'émetteur tire aléatoirement $s \leftarrow \mathbb{Z}_p^*$, et calcule :

- $k = H_1(e(g,h)^s)$,
- $C_1 = \left(g^{\alpha} g^{H_2(id)}\right)^s$,
- $C_2 = Enc(k, M)$.

Le chiffré à envoyer est :

$$C = (C_1, C_2)$$

Déchiffrer(sk_{id}, C) **:** En recevant $C = (C_1, C_2)$, le récepteur calcule :

- $k = H_1(e(C_1, sk_{id}))$,

Et il récupère le message comme suit :

$$M = Dec(k, C_2)$$

Preuve de fonctionnement

La justesse de la construction est assurée par l'équation ci-dessous qui permet au récepteur de récupérer la clé :

$$\begin{aligned} e(C_1, sk_{id}) &= e\left(\left(g^{\alpha} g^{H_2(id)}\right)^s, h^{\frac{1}{\alpha + H_2(id)}}\right) \\ &= e\left(g^{(\alpha + H_2(id))s}, h^{\frac{1}{\alpha + H_2(id)}}\right) \\ &= e\left(g,h\right)^{\frac{(\alpha + H_2(id))s}{\alpha + H2(id)}} \\ &= e(g,h)^s \end{aligned}$$

5.3 Implémentation

Nous avons préféré implémenter nos propres bibliothèques concernant les courbes elliptiques ainsi que le couplage et ne pas travailler avec des bibliothèques déjà exis-

tantes, afin de pouvoir réaliser notre schéma IBE.

Nous notons que tous les résultats du temps d'exécution sont donnés en seconde dans cette section et sont obtenus utilisant une machine qui possède un processeur « Intel Core i7-6200u up to 3.1GHz » et une RAM de 8GB, il est noté aussi que tous les paramètres utilisés sont en hexadécimal.

5.3.1 Implémentations existantes de couplage

Nous allons lister ci-dessous quelques implémentations déjà existantes du couplage :

- Magma [37] est une bibliothèque qui permet d'effectuer le calcul de couplage, elle a connu beaucoup d'évolutions à travers le temps et aujourd'hui elle permet le calcul des couplages Weil, Tate et quelques unes de leurs variantes.
- PBC [38] fut la toute première bibliothèque de calculs de couplages optimisé, introduite par Lynn, elle est écrite en C. Cependant elle n'a pas pu offrir les performances désirés.
- Miracl [39] est une bibliothèque performante, elle est développée en C++ par Michael Scott et son groupe de recherche.
- Relic [40] est une autre librairie codée en C++ par Diego Aranha et ses collaborateurs, elle est considérée comme une bibliothèque optimale en matière de calculs. Contrairement à Miracl, Relic est une librairie gratuite.
- DBPL [41] est une bibliothèque développée en 2018 par Kamel Mohamed Faraoun en langage Delphi, elle permet le calcul du couplage sur plusieurs familles de courbes adaptées où elle marque de bons résultats en matière de temps de calcul.

Il existe d'autres bibliothèques, mais nous préférons citer les plus importantes historiquement.

5.3.2 Outil de programmation utilisé

S'inspirant du langage de programmation *ABC*, En 1991, Guido van Rossum lança la première version de Python, un langage de programmation interprété.

Python est un langage dynamiquement typé, c'est-à-dire que le type des variables n'a pas besoin d'être spécifier. En plus, ce langage est facile à lire et à comprendre favorisant ainsi l'efficacité aux règles qui contrôlent sa syntaxe. Aussi, il est considéré comme étant un langage avec de multiple paradigme tel que l'impérative et l'orienté

objet, il propose une variété de bibliothèques permettant une gestion de mémoire automatique et des fonctionnalités dynamiques.

Tous ces facteurs mais encore notre curiosité à connaître et à interagir avec ce langage, nous ont mené à choisir Python comme langage pour l'implémentation de notre projet.

5.3.3 Partie 1 : Implémentation du couplage

La partie d'implémentation du coulage est la partie la plus difficile et complexe à réaliser, vu que beaucoup d'arithmétique et d'algorithmique se cachent derrière cette application bilinéaire, en plus, le domaine de recherche couvrant le couplage est très vaste et actif et touche beaucoup de domaines, on trouve aussi beaucoup d'optimisations pour alléger le calcul du couplage qui apparaît couteux.

Nous allons diviser la partie 1 en des étapes comme suit :

Étape 1

Nous avons d'abord implémenté les courbes elliptiques proprement dites tel que présentées dans le chapitre 2 puis qu'elles construisent les groupes qui définissent le couplage, nous avons d'abord commencer par implémenter les corps finis $\mathbb{F}_p$ avec toutes leurs arithmétiques, après nous avons utilisé les paramètres du domaine définit dans [42] pour définir une courbe elliptique le sextuplet (p, a, b, G, r, h) tel que p est un entier premier très grand qui définie $\mathbb{F}_p$, a et b sont des éléments de $\mathbb{F}_p$ qui définissent la courbe, G est un point générateur de la courbe, n est un premier qui présente l'ordre du point G et h est un cofacteur entier $h = \#E(\mathbb{F}_p)/n$.

Nous avons ensuite réussit à réaliser tous les algorithmes présentés dans le chapitre 2 tel que :

- Les différentes méthodes d'addition de points (addition affine, addition Jacobienne et addition projective), le tableau 5.1 montrent une comparaison entre les méthodes citées en terme de temps d'exécution d'addition et de doublement, la courbe utilisée dans la comparaison est la courbe « secp256k1 » définie dans [17] comme suit :
  p = fffffffe0000000100000000000000000000000ffffffffffffffffffffffff
  a = fffffffe0000000100000000000000000000000ffffffffffffffffffffffffc
  b = 5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604

b

x_G = 6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296

y_G = 4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5

r = ffffffff00000000ffffffffffffffffbce6faada7179e84f3b9cac2fc632551

h =1

Supposant $k = 3df2$ un entier tiré aléatoirement, pour le doublement on a utilisé la boucle définie comme suit :

$$E_i = \begin{cases} E_0 = G \\ E_i = 2E_{i-1} \end{cases}$$

Nous avons calculé $E_k = (x_{E_k}, y_{E_k})$ (c-à-d k doublement) tel que :

x_{E_k} = a7ce470b9bd437ded14caea7d6935fe5ba7562b0c88be08ad43e2ff46371026d

y_{E_k} = b3aabefbfff0438239b330367173929a1893a1ca221800006461aeb3d486f195

Et pour l'addition nous avons a utilisé la boucle définie comme suit :

$$F_i = \begin{cases} F_0 = G \\ F_i = F_{i-1} + G \end{cases}$$

On a calculé $F_k = (x_{F_k}, y_{F_k})$ (c-à-d k addition) tel que :

x_{F_k} = 26b00e3d4797eaf5778d3b671949f7881948abe52fe79e5eb93ef6842298338d

y_{F_k} = 454aa786c9cccc611a6b0ccd099f4de60fdcca1aa109e909daf67c61d856d96c

Les résultats pratiques valident la comparaison théorique du tableau 2.1 où les deux méthodes Jacobienne et projective marquent une supériorité par rapport à la méthode affine.

	Temps de doublement	Temps d'exécution d'addition
Addition affine	1.813218593597412	1.76517915725708
Addition Jacobienne	0.5154716968536377	0.48425769805908203
Adition projective	0.4855935573577881	0.390535831451416

Table 5.1 – Comparaison entre les méthodes d'addition de points

- Les algorithmes de multiplication par un scalaire (l'algorithme Double and Add et l'algorithme Addition-Substraction), le tableau 5.2 montre une comparaison entre les deux algorithmes en terme de temps d'exécution où on remarque l'avantage de l'algorithme Addition-Substraction par rapport à l'algorithme Double and Add, on note qu'on a utilisé comme méthode d'addition et de doublement de base la méthode affine qui n'est pas la plus rapide, on

a aussi utilisé la même courbe « secp256r1 » présentée ci-dessus et la multiplication qu'on a fait est : $G * k$ avec k un élément de $\mathbb{Z}_p$ tiré aléatoirement comme suit :
k = c1926ab7174c8ebf0a3eb1855cb7ae71f7e5c3ec1c4ff3071a6a47dc73f8c852
Le résultat de cette multiplication est $G^{'} = (x_{G'}, y_{G'})$ tel que :
$x_{G'}$ = 75eb6625967c85dc669b7f4abdb42b66744d8ef2ae9ba9e30a311e6dd01327c0
$y_{G'}$ = 4a2f48c1b8ee9ba7b193808234812a905b00bc6e21342eda79d9670652355946

	Temps d'exécution
Algorithme Double and Add	0.046894073486328125
Algorithme Addition-Substraction	0.031239748001098633

Table 5.2 – Comparaison entre les algorithmes de multiplication de point par un scalaire

Nous avons implémenté toutes les courbes elliptiques standardisées définies dans [17] qui sont considérées comme sécurisées pour usage sensible et qui sont en effet largement utilisées aujourd'hui à travers le monde.

Tous cela nous à permit à la fin de réaliser le système d'échange de clés ECDHKE présenté dans 2.6.2.

Étape 2

Ensuite, nous avons implémenté les extensions des corps finis présentés dans 2.2.5, nous avons réaliser les extensions de corps de la forme : $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^6}$ et $\mathbb{F}_{p^{12}}$ qui sont les extensions nécessaires pour construire la famille des courbes elliptiques compatible avec le couplage BN [24] présentée dans 3.9.3, nous avons choisit d'implémenter cette dernière. Les extensions de corps citées sont réalisées avec tous leurs algorithmes nécessaires tel que les opérations arithmétiques (+, -, ×, /, la puissance, la racine carré, ...etc) et d'autres fonctions pour manipuler et optimiser le couplage tel que la fonction de Frobenius [43] appliquée aux éléments de $\mathbb{F}_{p^{12}}$.

L'extension de corps $\mathbb{F}_{p^2}$ nous a permit de réaliser la courbe elliptique $E(\mathbb{F}_{p^2})$ qui est utilisé pour construire la famille de courbes BN.

Et maintenant que toutes les pièces nécessaires sont disponibles pour construire

des courbes de la famille BN, nous avons implémenté et utilisé les courbes BN définies dans [44], on note qu'une courbe BN (une courbe adaptée au couplage en général) a un nombre important de paramètres, on recommande au lecteur de voir [44] où ils sont expliqués.

Notre bibliothèque est capable de générer toute une courbe BN seulement passant le paramètre u présenté dans 3.9, mais par convention et aussi pour éviter le temps de génération de la courbe nous avons permit la possibilité de mettre les paramètres des courbes manuellement tel qu'elles sont présentés dans [44].

Étape 3

Dans cette étape, nous avons implémenté l'algorithme de Miller tel qu'il est présenté dans l'algorithme 3.1, on rappelle que ce dernier calcule une fonction rationnelle de type $f_{s,P}(Q)$, cet algorithme se base sur une boucle qu'itère sur la représentation binaire d'un entier ou sur sa représentation NAF présentée dans 2.4.5.

Nous avons en parallèle implémenté l'exponentiation finale dont on a parlé brièvement dans le chapitre 3, cette dernière est utilisée dans le couplage de Tate et ses variantes et elle assure les résultats identiques de ces couplages (propriétés de bilinéarité).

Le couplage de Tate se constitue de deux étapes, algorithme de Miller et exponentiation finale, on a réussit donc à implémenté ce couplage ainsi que ses variantes présentées dans 3.8 tel que Ate et Optimal-Ate, ce dernier a nécessité l'implémentation d'autres fonctions comme nous l'avons présenté. Rappelons qu'on a appliqué ces couplages sur la famille BN que nous avons construit dans l'étape 2.

Le tableau 5.3 présente une comparaison entre le couplage de Tate et ses variantes : Ate et Optimal-Ate en terme de temps d'exécution où on remarque que Optimal-Ate atteint les meilleurs résultats. On a utilisé dans la comparaison la courbe BN « BEUCHAT256 » définie dans [44] dont les deux point P et Q passés en entré du couplage sont les générateur de $\mathbb{G}_1$ et $\mathbb{G}_2$ respectivement tel que la courbe est définie.

	Temps d'exécution
Tate	0.265566349029541
Ate	0.15624237060546875
Optimale-Ate	0.10935163497924805

Table 5.3 – Comparaison entre les couplages Tate, Ate et Optimal-Ate en terme de temps d'exécution

Pour montrer la propriété de la bilinéarité présentée dans 3.2, nous avons choisi a et b aléatoirement et assurer que l'égalité $e(aP, bQ) = e(bP, aQ) = e(P, Q)^{ab}$ est vérifiée.

5.3.4 Partie 2 : implémentation de l'approche IBE proposée

En plus de l'opération de la multiplication d'un point d'une courbe elliptique par un scalaire et le couplage, nous avons vu que notre schéma IBE nécessite deux fonctions de hachage cryptographiques et un chiffrement symétrique authentifié. Nous avons implémenté la fonction de hachage H_1 à base de la fameuse fonction de hachage SHA256 et la fonction de hachage H_2 à base de fonction de hachage SHA512. Les deux fonctions de hachage SHA256 et SHA512 existent déjà dans Python sous la package « Hashlib ». Comme un chiffrement authentifié nous avons préféré de travailler avec AES256, l'un des plus forts cryptosystèmes symétriques connus à nos jours et qui existe déjà dans le fameux package « PyCryptodome », ce dernier offre un nombre importants de primitives cryptographiques.

Après avoir préparer toutes les pièces nécessaires, nous avons mis en œuvre notre schéma IBE, nous avons donc implémenté les quatre algorithmes définissants notre schéma : Initialiser, Extraire, Chiffrer et Déchiffrer.

Pour valoriser notre travail et pour faire simuler notre schéma en réalité on a déployé notre schéma dans un réseau, pour cela nous avons utilisé les deux modules de Python : « Socket » qui gère la communication entre des hôtes en utilisant un couple (Adresse IP, Port) pour identifier un hôte et « Pickle » qui gère l'encapsulation des données de tout type pour assurer leur bon transfert.

5.4 Résultats et discussion

Comme nous l'avons déjà motionné, notre schéma se base sur la construction de Wee [29], nous avons travaillé avec un couplage asymétrique où on a utilisé des groupes bilinéaires d'ordre premier au lieu des groupes bilinéaires d'ordre composé, en plus notre schéma offre un chiffrement de message d'une taille quelconque, une propriété qui n'est pas disponible dans la plus part des schémas IBE existants, on a garantie cette propriété en utilisant un chiffrement authentifié symétrique.

En comparant notre schéma avec le schéma de Wang et Zhao [34], les deux schémas se ressemblent dans le fait d'utiliser un chiffrement symétrique authentifié, mais il se différent dans la technique de génération de la clé. Leur schéma offre en plus du chiffrement, la signature numérique du message.

Notre construction marque des avantages en terme de complexité par rapport à plusieurs autres schémas, elle utilise un couplage dans l'initialisation et un couplage dans le déchiffrement, ce qui n'est pas le cas dans le travail de Watanabe, Emura et Seo [32] qui ont utilisé un couplage dans l'initialisation et trois couplages dans le déchiffrement et le travail de Karatsuma et Yamada [31] qui ont utilisé un couplage dans le chiffrement et plusieurs couplages dans le déchiffrement par exemple.

On présente ci-dessous un exemple numérique de notre schéma IBE en appliquant les quatre algorithmes comme suit :

Initialisation(λ) :
- on utilise la courbe de la famille BN « BEUCHAT256 » présentée dans [44] avec tous ses paramètres, elle est générée par le paramètre u = 3fc0100000000000,
- on utilise le couplage Optimal-Ate,
- on utilise le générateur $g = (x_g, y_g)$ de $\mathbb{G}_1$ de la courbe « BEUCHAT256 » tel que :
  $x_g = 1$
  y_g = d45589b158faaf6ab0e4ad38d998e9982e7ff63964ee1460342a592677cccb0
- on définit la fonction $H_1 : \mathbb{G}_T \to \{0,1\}^{256}$ comme suit : H_1 = SHA256($X \in \mathbb{G}_T$),
- et on définit la fonction $H_2 : \{0,1\}^* \to \{\{0,1\}^{512} \bmod p\}$ comme suit : H_2 = SHA512($X \in \{0,1\}^*$) mod p.

Les aléatoires que nous avons tiré pour l'exemple sont :

- une clé master secrète
  α = 134be0d30c819d2cf8b2d963eeb81b7cd3061e16e6eb5b4417fb0434351d9727
- un élément de $\mathbb{G}_2$: $h = (x_h, y_h)$ tel que
  x_h = (1c0bf6e00b6ff3e841f833f28387c609603d86708528bdd36d7f358399e34d44 + 157dabb91e31e0dadefd9bc9661e9bd5d01e648c4bb6fca30966897cb83a4498β)
  y_h = (1fb0396da809aff647794d0c9bca5ba5b6019dd2289eaeea6cdf9ba415a2a1b1 + 1f0fee7495e958e3163fa83bb52b18f8d9381ff479096a4487de711f254eac41β)

On calcule les autres paramètres publiques :

- l'élément $g^\alpha = (x_{g^\alpha}, y_{g^\alpha})$ tel que
  x_{g^α} = 1d1a1b4412d1799fb4f366a9aa9f97fe7afe21e6c521d20f80d40772e3330d79
  y_{g^α} =1a3cccf6356c78a8381bcd714bbac86b64322da26fdc3ea627a733238fcd3e13
- le couplage $e(g, h)$ = (((127e16047c0c7c41ff5d680b21a92e6641d2d23dbf83db563f1290b1d74c5e30 + 157a02c9b9a31bb994ad47742df766d32f7e7a851fa9998599534e48ccb162d5β) + (100bc394d6bf65d5a15cabbd65850c3ace2ee986e6b06984f805376429e5b12 + f9e1257c6d47da1b59774bcd49e73672f9a64f1875d57e7f0409a8b55d44b06β)γ + (12f900acc07232abe21d14e7cc683b3a5771fc34e11302353199b734bd2d71e3 + e54ef59ae70283d8b6723300076c034fe7d9194cb297d67058f0b4d379f4229cβ)γ^2) + ((206c72b34c80fa9fc4b6452bd52f701422f36eca8022bcc8108608dea66d7e23 + c6cea87dac9519aa5b1b6010a022bdade2a3adb926f1ba36ba58acad4d3e323β) + (f055d354233764947290122c13c6bcc9e9d76565a0fa6dfefcf702b48ebcf04 + 1c6c14ed4f368a3b6111294ab983e631fea32221f8939f298f339a4444cdbf83β)γ + (e2e1fc038e96cf486612b33bb3007c06ad55a8bc05adf6bd5e778112d13935a + 1a981cd746e66d58c1b8ed7d4786ab53a880dbe2cbdb6905fdd1a0f360337562β)γ^2)δ)

Extraction(msk, id) : Suposant l'id du recepteur est « Abdelhakim@email », le PKG génère la clé secrète associée à cette id comme suit :

$$sk_{id} = h^{\frac{1}{\alpha + H_2(id)}} = (x_{sk_{id}}, y_{sk_{id}}) \text{ tel que :}$$

$x_{sk_{id}}$ = (1b36a05320a5b86c9cb54c329625bc047a9138c3a55e5d05cf99d5f4c2d03b42 + fa08ab7bcf0e1ae6383569b2fc952436c44e51011b5f4bbd45218132cde9e1eβ)

$y_{sk_{id}}$ = (175f231a114041732f14d557f993b4315f4328bc96d769172dfadc823bf98 fa6 + 19ab12d39777f29f5899b23faeae13eebd6681ca6e65afa15af2f70e1ca6327 1β)

Chiffrer(mpk, id, M) **:** En utilisant le chiffrement authentifié sécurisé AES256 avec le mode opératoire sécurisé CBC et en utilisant un vecteur d'initialisation tiré aléatoirement pour l'exemple IV= 81c8e24a3951b075800940fb166a9fbd, et suposant le message à chiffrer est :
M = « ***Je suis un message claire à chiffrer, je peux être d'une taille quelconque.*** ».
L'élément s de $\mathbb{Z}_p^*$ aléatoirement tiré pour l'exemple est :
s = ec813ac6598b6943b9c8a875e8eeadc03b18382c0d216456499210fac7b5fd1,
l'émetteur calcule :

- $k = H_1\left(e(g,h)^s\right)$ = 49c50cbafe95d2778f3e393ba3f5f1323aab0a680f025aef 149791e3704f95af
- $C_1 = \left(g^{\alpha} g^{H_2(id)}\right)^s = (x_{C_1}, y_{C_1})$ tel que :
  x_{C_1} = 3cde2eea98ec53e4f8079b67a36edcead3de5e311fb11f8d1efdd31482b 1601
  y_{C_1} = 8ad1845e9fa10798e37665822acdca1ae584d317973f5f7c1341ad8b527 21cb
- $C_2 = Enc_{AES256}(k, M, \text{IV})$ = 57e96d0bec88e3e3304745acebd32fe978a96e0 b1dd04b3d9b191b355171773801fe5c35edcd1a394150a4e0d35ae9dffdde2d2 69c444b76cfea1ce8639dc5162caaf070e613c4e078c48a1122b2aab9

Le chiffré à envoyer est :

$$C = (C_1, C_2)$$

On note aussi qu'on doit envoyer le vecteur d'initialisation IV publiquement au récepteur.

Déchiffrer(sk_{id}, C) **:** En recevant $C = (C_1, C_2)$, le récepteur calcule :

- $k = H_1(e(C_1, sk_{id}))$ = 49c50cbafe95d2778f3e393ba3f5f1323aab0a680f025a ef149791e3704f95af

Et il récupère le message comme suit :

$M = Dec_{AES256}(k, C_2, \text{IV})$ =« ***Je suis un message claire à chiffrer, je peux être d'une taille quelconque.*** »

Tableau 5.4 présente le temps d'exécution pour chaque algorithme de l'exemple précédent.

	Temps d'exécution
Initialisation	0.34366822242736816
Extraction	0.23429155349731445
Chiffrer	0.10934662818908691
Déchiffrer	0.10934972763061523

Table 5.4 – Temps d'exécution pour chaque algorithme d'exemple du notre schéma IBE

5.5 Conclusion

Nous avons proposé dans ce chapitre un schéma IBE en utilisant le couplage avec des groupes bilinéaires d'ordre premier, nous avons présenté son principe et son fonctionnement ainsi que la preuve de son fonctionnement, nous avons aussi présenté les différentes étapes d'implémentation du couplage et du schéma proposé, nous avons à la fin illustrer notre travail avec un exemple de chiffrement de notre proposition. Notre schéma proposé marque quelque avantages que nous avons cité.

Chapitre 6

Conclusion générale

À travers ce travail, nous avons pu couvrir les différents points et questions soulignés dans l'introduction, nous proposons un schéma de chiffrement à base d'identité, sous l'hypothèse du problème SXDH. Notre construction est basée sur la construction de Wee [29] , permettant une sécurité IND-ID-CPA et avec notre propre implémentation du couplage, qui nous a permis d'obtenir de bons résultats.

Nous pourrons optimiser encore plus le temps de calcule du couplage, rendant ainsi le chiffrement et le déchiffrement de message de taille importante plus fluide et chercher à remplacer l'oracle aléatoire en exploitant d'autres alternatives par rapport au problèmes mathématiques utilisés.

En plus des quelques pistes citées ci-dessus, nos perspectives visent à :

- Renforcer notre système en intégrant les schémas de signature courte,
- découvrir et utilisé « les réseaux euclidiens (Lattice en englais) », une primitive mathématique qui commence à prendre sa place dans la cryptographie ces dernières années et qui est aussi utilisée pour concevoir des cryptosystèmes à base d'identité.

Enfin nous souhaiterons construire notre propre bibliothèque de couplage en Python à partir de notre implémentation qui fonctionne pour le moment avec les courbes BN mais qui pourrai fonctionner avec toute autre famille de courbes adaptées au couplage.

Pour terminer, notre travail fait l'objet d'une soumission en cours d'un article à la conférence CITCS'2019 (Conference on Innovative Trends in Computer Science) qui se tiendra à Guelma en novembre 2019.

Bibliographie

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

[2] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976.

[3] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985.

[4] Antoine Joux. A one round protocol for tripartite diffie–hellman. In *International algorithmic number theory symposium*, pages 385–393. Springer, 2000.

[5] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the theory and application of cryptographic techniques*, pages 47–53. Springer, 1984.

[6] Christof Paar and Jan Pelzl. *Understanding cryptography : a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[7] Naomi Benger and Michael Scott. Constructing tower extensions of finite fields for implementation of pairing-based cryptography. In *International Workshop on the Arithmetic of Finite Fields*, pages 180–195. Springer, 2010.

[8] Selçuk Baktir and Berk Sunar. Optimal tower fields. *IEEE Transactions on Computers*, 53(10) :1231–1243, 2004.

[9] Nadia El Mrabet and Marc Joye. *Guide to Pairing-Based Cryptography*. Chapman and Hall/CRC, 2017.

[10] Alka Sawlikar. Point multiplication methods for elliptic curve cryptography. *International Journal of Engineering and Innovative Technology (IJEIT)*, 1(1) :1–4, 2012.

[11] Dan Boneh and Victor Shoup. *A graduate course in applied cryptography*. 2016.

[12] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer Science & Business Media, 2009.

[13] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[14] Antoine Joux and Kim Nguyen. Separating decision diffie–hellman from computational diffie–hellman in cryptographic groups. *Journal of cryptology*, 16(4) :239–247, 2003.

[15] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001.

[16] Alfred J Menezes, Tatsuaki Okamoto, and Scott A Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *iEEE Transactions on information Theory*, 39(5) :1639–1646, 1993.

[17] Daniel RL Brown. Sec 2 : Recommended elliptic curve domain parameters. *Certicom Research*, 2010.

[18] Gerhard Frey, Michael Muller, and H-G Ruck. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45(5) :1717–1719, 1999.

[19] Ben Lynn. *On the implementation of pairing-based cryptosystems*. PhD thesis, Stanford University Stanford, California, 2007.

[20] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985.

[21] Florian Hess, Nigel P Smart, and Frederik Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10) :4595–4602, 2006.

[22] Frederik Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1) :455–461, 2009.

[23] Paulo SLM Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *International Conference on Security in Communication Networks*, pages 257–267. Springer, 2002.

[24] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *International Workshop on Selected Areas in Cryptography*, pages 319–331. Springer, 2005.

[25] Ezekiel J Kachisa, Edward F Schaefer, and Michael Scott. Constructing brezing-weng pairing-friendly elliptic curves using elements in the cyclotomic field. In *International Conference on Pairing-Based Cryptography*, pages 126–135. Springer, 2008.

[26] Craig Costello. *Fast formulas for computing cryptographic pairings*. PhD thesis, Queensland University of Technology, 2012.

[27] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Annual International Cryptology Conference*, pages 443–459. Springer, 2004.

[28] Brent Waters. Dual system encryption : Realizing fully secure ibe and hibe under simple assumptions. In *Annual International Cryptology Conference*, pages 619–636. Springer, 2009.

[29] Hoeteck Wee. Déjà q : Encore ! un petit ibe. In *Theory of Cryptography Conference*, pages 237–258. Springer, 2016.

[30] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *Theory of Cryptography Conference*, pages 455–479. Springer, 2010.

[31] Shuichi Katsumata and Shota Yamada. Partitioning via non-linear polynomial functions : more compact ibes from ideal lattices and bilinear maps. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 682–712. Springer, 2016.

[32] Yohei Watanabe, Keita Emura, and Jae Hong Seo. New revocable ibe in prime-order groups : adaptively secure, decryption key exposure resistant, and with short public parameters. In *Cryptographers' Track at the RSA Conference*, pages 432–449. Springer, 2017.

[33] Charanjit S Jutla and Arnab Roy. Shorter quasi-adaptive nizk proofs for linear subspaces. *Journal of Cryptology*, 30(4) :1116–1156, 2017.

[34] Hongbing Wang and Yunlei Zhao. Identity-based higncryption. 2019.

[35] Paulo SLM Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In *International conference on the theory and application of cryptology and information security*, pages 515–532. Springer, 2005.

[36] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 56–73. Springer, 2004.

[37] Wieb Bosma, John Cannon, and Catherine Playoust. The magma algebra system i : The user language. *Journal of Symbolic Computation*, 24(3-4) :235–265, 1997.

[38] Ben Lynn et al. Pbc library. *Online : http ://crypto.stanford.edu/pbc*, 59 :76–99, 2006.

[39] Michael Scott. Miracl library. indigo software, 2005.

[40] D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `http://code.google.com/p/relic-toolkit/`.

[41] Faraoun Kamel, Mohamed. Delphi's bilinear pairings library. `https://github.com/kamel78/DBPL`, 2018.

[42] Daniel RL Brown. Sec 1 : Elliptic curve cryptography. *Certicom Research*, 2009.

[43] Jean-Luc Beuchat, Jorge E González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In *International Conference on Pairing-Based Cryptography*, pages 21–39. Springer, 2010.

[44] Akihiro Kato, Michael Scott, Tetsutaro Kobayashi, and Yuto Kawahara. Barreto–naehrig curves. 2016. `https://tools.ietf.org/html/draft-kasamatsu-bncurves-02`.