Université de Montréal

# Cryptage de Messages SMS

Par

Tayeb Zebiche

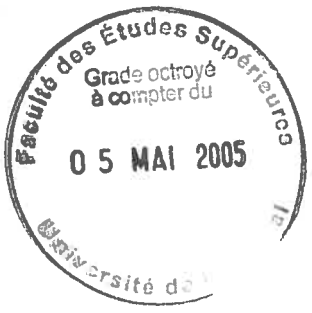
Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maîtrise ès sciences (M.Sc)
en informatique

Décembre, 2004





Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

**Cryptage de Messages SMS**

présenté par:

Tayeb Zebiche

a été évalué par un jury composé des personnes suivantes:

Alain Tapp
(Président-rapporteur)

Stefan Wolf
(Directeur de recherche)

Peter Kropf
(Co-directeur)

Gilbert Babin
(Membre du jury)

Mémoire accepté le:

15 mars 2005

# Sommaire

Ce mémoire propose la conception et la réalisation d'un système de chiffrement de la messagerie texte mobile (SMS). Nous utilisons un système de chiffrement basé sur l'identité (IBE) où le numéro de téléphone mobile (l'identité) du destinataire est utilisé comme étant une clé publique de chiffrement. Le système IBE utilisé n'exige aucune certification de clés publiques ou échange de clés secrètes, du moment où les identités des usagers sont des connaissances publiques. Lorsqu'une identité est concaténée avec le mois en cours, les clés privées seront valides uniquement pendant ce mois-ci, facilitant ainsi leur révocation. Ce mode de chiffrement implique l'utilisation d'une autorité de génération de clés privées (PKG) dont le rôle se limite à la génération des clés privées qui sont nécessaires pour le déchiffrement.

Vu l'importance des couplages bilinéaires dans la construction des systèmes IBE, nous avons conduit une étude mathématique, accompagnée d'exemples concrets, qui explique la construction du couplage de Tate réduit. Ce projet inclut l'implantation du système en Java 2 Micro Edition (J2ME). D'une taille d'environ 55 ko, la MIDlet (application J2ME) développée peut être exécutée sur les téléphones mobiles dotés de la plate-forme J2ME. Les délais de chiffrement/déchiffrement sont d'environ 7 secondes sur un émulateur de Nokia 6230 pour une taille de groupe de l'ordre de 360 bits.

La sécurité du système est comparable à celle du cryptosystème ElGamal dans $\mathbb{F}_p$. Nous supposons qu'il existe un canal sécuritaire pour acheminer les MIDlets, accompagnées des clés privées associées, aux usagers des téléphones mobiles.

Ce travail est inspiré des travaux de Boneh et Franklin [BF01] et Barreto et al. [BKLS02], [BLS03].

**Mots Clés :** SMS, Chiffrement basé sur l'identité, Couplage de Tate, Algorithme de Miller, Générateur de clés privées, MIDlet.

# Abstract

This thesis focuses on the conception and the realization of an SMS cryptosystem. We use an Identity-Based Encryption (IBE) System, where the recipient's mobile-phone number is used as the public key of the encryption. Such IBE systems do not require any public-key certification or secret-key exchange, because identities of users are public knowledge. When an identity is concatenated with the current month, then the generated private keys will be valid only during this month, thus facilitating their revocation. IBE systems involve one third party called Private Key Generator (PKG), whose role is limited to generate private keys that are needed for decryption.

Considering the importance of bilinear pairings in the construction of IBE systems, we have conducted a mathematical study, accompanied by concrete examples, which explains the reduced Tate pairing construction. This project includes a full J2ME implementation of the target IBE cryptosystem. The MIDlet (J2ME application) size is about 55 KBytes and can be executed on J2ME devices such as mobile phones. Encryption/decryption time is about 7 seconds on Nokia 6230 emulator for a group size of 360 bit-length.

The system's security is comparable to the one of the ElGamal cryptosystem in $\mathbb{F}_p$. We assume that there exists a secure channel to send MIDlets, as well as private keys, from the PKG to their corresponding user's mobile phones.

This study is inspired by works of Boneh and Franklin [BF01] and Barreto et al. [BKLS02], [BLS03].

**Keywords :** SMS, Identity-based encryption, Tate Pairing, Miller's algorithm, Private Key Generator, MIDlet.

# Table des Matières

# Liste des tableaux

# Table des figures

# Liste des Algorithmes

# Liste des abréviations

| | |
|---|---|
| AC | Autorité de Certification |
| BDH | Bilinear Diffie-Hellman |
| CDH | Computational Diffie-Hellman |
| CLDC | Connected Limited Device Configuration |
| CRL | Certificate Revocation List |
| DDH | Decisional Diffie-Hellman |
| DL | Discrete Logarithm |
| ECDLP | Elliptic Curve Discrete Logarithm Problem |
| GDH | Gap Diffie-Hellman |
| IBE | Identity-Based Encryption |
| ID-PKC | Identity-Based Public Key Cryptography |
| IND-CCA | Indistinguishability Against Chosen Ciphertext Attack |
| IND-CPA | Indistinguishability Against Chosen Plaintext Attack |
| MIDP | Mobile Information Device Profile |
| PKC | Public Key Cryptography |
| PKG | Private Key Generator |
| PKI | Public Key Infrastructure |
| WMA | Wireless Messaging API |

# Remerciements

Je remercie en premier lieu ma femme Khedidja qui m'a soutenu moralement et financièrement durant toutes ces années.

Je tiens à remercier Stefan Wolf, mon enseignant et directeur de recherche, pour m'avoir fait découvrir le monde de la cryptographie, pour ses directives et conseils précieux durant la rédaction de ce mémoire.

Je tiens également à remercier Peter Kropf, mon codirecteur, pour avoir suivi ce travail et pour ses conseils et encouragements tout au long de ce projet.

Je remercie tout particulièrement les membres de mon jury de mémoire, qui ont accepté de juger ce travail. J'adresse mes très sincères remerciements à Alain Tapp pour avoir présidé le jury ainsi que pour ses remarques constructives.
Je remercie également Gilbert Babin, j'ai beaucoup apprécié sa participation au jury de ce mémoire.

Merci à tous ceux qui ont pris part à la lecture et la correction de ce document, Soheil Chennouf, Ahmed Louzani, Lamri Benhafid, Azzedine Djeraba, Rachid Chebanni et son épouse Salima Touahria.

*À mon fils Adlène.*

# Chapitre 1

# Introduction

SMS (Short Message Service) est un service qui permet d'envoyer et de recevoir sur un téléphone mobile GSM[1] un message court d'au plus 160 caractères. Le texte peut être composé de caractères alphanumériques [SL00].

Le réseau GSM est capable de sauvegarder les message SMS pour un acheminement différé. Cela veut dire que si le téléphone mobile destinataire est éteint ou hors de la zone de couverture, les messages qui lui sont destinés sont sauvegardés de telle sorte qu'on peut les recevoir lors de la mise en marche de l'appareil.

La messagerie texte mobile a connu une croissance fulgurante durant ces dernières années. Partant du constat que le téléphone mobile nous accompagne partout, on assiste à une nouvelle forme de commerce électronique appelé commerce mobile ou *m-commerce*. Plusieurs banques à travers le monde utilisent SMS pour gérer une partie de leurs services. Par exemple, les clients peuvent consulter leurs bilans bancaires via SMS [RLB04]. Par ailleurs, France Télécom et le groupe Transdev sont en train

[1] Global System for Mobile communications

d'expérimenter un service de billet de bus électronique accessible sur le téléphone mobile en utilisant le SMS [Tél].

Vu la popularité de SMS, Java 2 Micro Edition (J2ME) fournit une API de messagerie sans-fil WMA[2] qui permet aux applications des appareils mobiles (eg. cellulaires, PDAs) d'envoyer et de recevoir des messages SMS. Ainsi, J2ME est une plateforme qui fournit un environnement standard pour développer des applications destinées pour les appareils mobiles [Yua04]. Malheureusement, WMA ne fournit aucun mécanisme de sécurité pour l'échange de messages SMS. D'autant plus que, SMS n'utilise pas le TCP comme un protocole de transport. Par conséquent, on ne peut pas utiliser HTTPS pour sécuriser la transmission de SMS.

## 1.1 Problématique

La messagerie texte mobile n'est pas un moyen de communication sécuritaire ; par défaut, le contenu de SMS est échangé sur le réseau GSM dans un format de texte clair [RLB04], [Mag02]. Ceci permet à un adversaire possédant les moyens appropriés d'attaquer les informations contenues dans ce message. Cela pourrait être tout simplement un employé malhonnête dans un centre de SMS (ou SMSC).

Deux défis majeurs sont à relever lors de la conception d'un système de chiffrement de SMS :

- le premier est la puissance de calcul limitée des CPU et la faible capacité de la

[2]Wireless Messaging API

mémoire des téléphones mobiles. Sachant que le système de chiffrement fait intervenir des opérations cryptographiques plus ou moins complexes, dépendamment du mode de chiffrement choisi ;

- le deuxième défi est lié à la limitation imposée sur la longueur d'un message SMS. Cela veut dire que l'opération de chiffrement pourrait ajouter des octets supplémentaires réduisant ainsi davantage le nombre de caractères pouvant être envoyés par un message SMS.

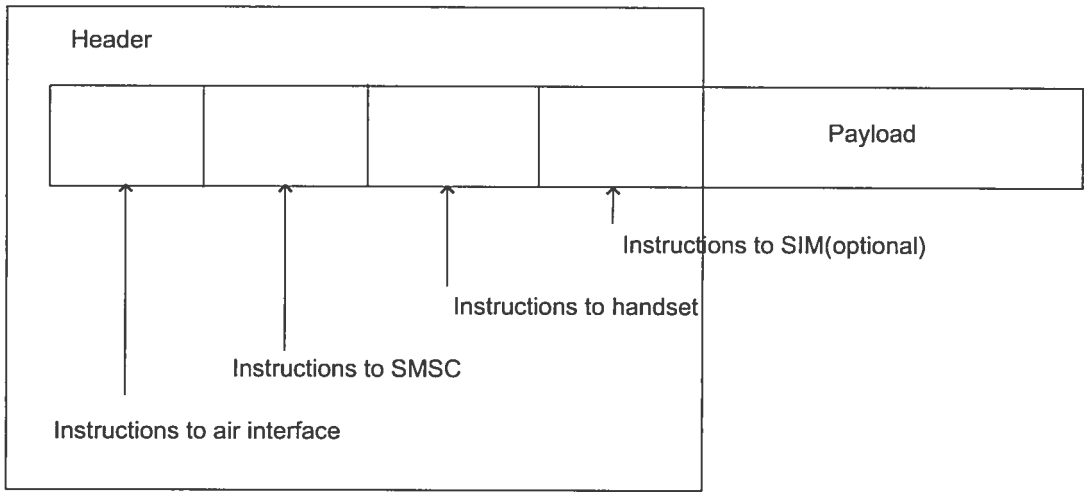



FIG. 1.1 – Structure d'un message SMS [Cle02]

## 1.2 Objectif

L'objectif de ce mémoire est la conception et la réalisation d'un système de chiffrement qui assure la confidentialité et l'intégrité des messages SMS échangés entre les téléphones mobiles dotés de la plate-forme J2ME. Nous voulons que le système

en question soit sécuritaire, efficace et simple à utiliser. On s'attend à ce que tous les pairs participants, utilisant ce système, puissent envoyer et recevoir des messages SMS sécurisés sans faire intervenir d'autres tiers ou participants, et sans utiliser de mots de passe.

## 1.3 Approches

La cryptographie traditionnelle nous enseigne qu'il existe généralement deux modes de chiffrements : symétrique et asymétrique. Le chiffrement symétrique (eg. DES) a l'avantage d'être rapide mais présente plusieurs lacunes. Entre chaque paire de participants, on prévoit une clé secrète pour pouvoir établir une communication sécuritaire. À titre d'exemple, dans un réseau comme internet ou SMS, le nombre de clés secrètes nécessaires pour $n$ participants est égal à $\frac{n(n-1)}{2}$ et toutes ces clés doivent être acheminées d'un bout à l'autre. On risque ainsi de perdre les clés et mettre en péril la sécurité du système.

Le chiffrement asymétrique, ou encore à clé publique, (eg. RSA, ElGamal) quant à lui, assure des communications sécuritaires. Par ailleurs, il est très lent [Yua04]. La préoccupation majeure reste la distribution et la certification des clés publiques.
Dans [RLB04], Ratshinanga et al. proposent un protocole d'échange de messages SMS sécuritaires de type client-serveur. Ce protocole combine les deux modes de chiffrement cités plus haut et nécessite trois envois de messages SMS pour l'ouverture d'une session client-serveur sécuritaire.

### 1.3.1 Notre démarche

Pour l'échange de messages SMS sécuritaires, nous proposons une solution d'appareil mobile à appareil mobile basée sur la cryptographie à clé publique basée sur *l'identité* qui a été proposée initialement par Shamir [Sch84] en 1984. Son but ultime était de soulager l'embarras des certificats qui sont essentiels dans les systèmes de chiffrement à clé publique conventionnels pour garantir l'authenticité des clés publiques. Cette idée demeurait un problème ouvert durant plusieurs années. En 2001, Boneh et Franklin [BF01] proposaient la première implémentation complète d'un système de chiffrement basé sur l'identité (IBE)[3], en utilisant *la bilinéarité des couplages.* Ce système consiste en quatre algorithmes : *setup*, *extract*, *encrypt* et *decrypt.*
Lorsque Alice veut envoyer un message $M$ à Bob en utilisant l'identité de ce dernier, le système IBE fonctionne comme suit :
*Setup :* Exécuté par l'autorité PKG[4], cet algorithme génère les paramètres publiques du système *params* et la clé maître $s$ qui demeure secrète chez le PKG.
*Encrypt :* Exécuté par Alice pour le chiffrement du message source $M$ en utilisant la clé publique (l'identité de Bob) $ID$ et les paramètres publiques *params*. Cela génère un cryptogramme $C$.
*Extract :* Exécuté par le PKG à la demande de Bob. Cela retourne la clé privée $d_{ID}$ de Bob associée à son identité $ID$.
*Decrypt :* Exécuté par Bob pour le déchiffrement du cryptogramme $C$ en utilisant sa clé privée $d_{ID}$ et les paramètres *params*.

Pour simplifier l'algorithme d'extraction des clés privées sur les appareils mobiles,

[3]Identity Based Encryption
[4]Private Key Generator

Boneh et al. [BF01] proposent la notion de *délégation d'une clé privée* où cette dernière demeure valide pendant une période bien déterminée. Par exemple, le jour, le mois ou l'année en cours. Aucune certification de clé publique n'est alors exigée. L'identité d'un participant est une information (une chaîne de caractères) qui l'identifie de manière unique et incontestable et qui est disponible publiquement [BF01], [Sch84]. Par exemple, une adresse de courriel, un numéro de téléphone, un numéro d'assurance sociale, ... etc.

Le système de chiffrement de messages SMS projeté utilise le numéro de téléphone mobile du destinataire comme étant une identité.

## 1.4 Organisation des chapitres

Dans le chapitre 2 nous présentons la théorie mathématique qui nous aidera à comprendre le calcul des couplages bilinéaires. Nous y trouveront en particulier la théorie des courbes elliptiques, les fonctions sur les courbes elliptiques, les diviseurs et les couplages bilinéaires de Weil et de Tate.

Le chapitre 3 est une suite logique du chapitre précédent, il est consacré au calcul des deux fameux couplages bilinéaires. Nous avons choisi le couplage de Tate pour sa rapidité de calcul.

Dans le chapitre 4, nous présentons les avantages et la sécurité des systèmes IBE en comparaison avec les systèmes de chiffrement à clé publique traditionnels. Nous concluons ce chapitre par la présentation du système IBE FullIdent que nous voulons implanter. Par la suite, le chapitre 5 se veut une étude de la messagerie SMS dans

la plateforme J2ME. Nous verrons en particulier l'envoi et la réception de messages SMS offerts par le paquetage javax.wireless.messaging.

Dans le chapitre 6, nous implantons le système IBE FullIdent à partir du couplage de Tate et la courbe supersingulière $E_{1,0}$. Cela est effectué en décrivant les principaux algorithmes cités plus haut. Nous finalisons ce chapitre par un exemple concret de fonctionnement du système pour l'envoi et la réception de message SMS sécurisés.

Le chapitre 7 est consacré à l'analyse de performances du système implanté. Cette analyse se focalise sur les délais de chiffrement/déchiffrement, la sécurité, et le nombre de caractères pouvant être envoyés par un message SMS crypté. Dans l'analyse de la sécurité du système, nous présentons le nombre d'opérations et le temps nécessaires pour qu'un adversaire, possédant des capacités calculatoires limités, réussisse une attaque relativement à des tailles de groupes variées. Par ailleurs, nous proposons quelques solutions pour améliorer l'efficacité du système.

Nous terminons ce mémoire par une conclusion et quelques recommandations pour d'éventuels futurs travaux.

# Chapitre 2

# Préliminaires

Dans ce chapitre, nous présentons une théorie mathématique dont nous avons besoin pour comprendre l'implémentation des cryptosystèmes basés sur les couplages bilinéaires. La première section est consacrée aux corps et extension de corps. Dans la deuxième section, nous traitons brièvement de la théorie des courbes elliptiques ; les fonctions sur les courbes elliptiques seront abordées dans la troisième section. Nous introduisons la théorie des diviseurs et le calcul de la fonction d'un diviseur principal dans les quatrième et cinquième sections respectivement. La dernière section est consacrée aux couplages de Weil et de Tate ainsi qu'à leurs propriétés respectives.

## 2.1 Corps Finis

### 2.1.1 Groupes

Un groupe $\mathbb{G}$ est un ensemble muni d'une loi de composition interne $\circ$ :

$$\mathbb{G} \circ \mathbb{G} \rightarrow \mathbb{G}$$

et possède les propriétés suivantes :

- Associative : $\forall a, b, c \in \mathbb{G}, a \circ (b \circ c) = (a \circ b) \circ c$.
- $\mathbb{G}$ a un élément neutre $e \in \mathbb{G}$, tel que $\forall a \in \mathbb{G}, a \circ e = e \circ a = a$.
- pour chaque élément $g \in \mathbb{G}$, il existe un élément inverse $g^{-1}$ tel que $g \circ g^{-1} = g^{-1} \circ g = e$ .

Le groupe $\mathbb{G}$ est Abélien (ou commutatif) si pour tout $a, b \in \mathbb{G}, a \circ b = b \circ a$. Le groupe est fini s'il contient un nombre fini d'éléments. Le nombre d'éléments dans un groupe fini est appelé l'ordre du groupe et est dénoté par $|\mathbb{G}|$. L'ordre d'un élément $a \in \mathbb{G}$ noté $ord(a)$ est le plus petit entier positif $i$ tel que

$$a^i = \underbrace{a \circ a \circ \cdots \circ a}_{i\,fois} = e.$$

$\mathbb{G}$ est *cyclique* s'il existe un élément g $\in \mathbb{G}$ , appelé un *générateur*, tel que pour tout élément $a \in \mathbb{G}$, il existe un entier $i$ tel que $a = g^i$. On écrit

$$\mathbb{G} = < g >$$

Deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ sont appelés *isomorphes*, $\mathbb{G}_1 \cong \mathbb{G}_2$ , s'il existe une application bijective $\Phi$ de $\mathbb{G}_1$ vers $\mathbb{G}_2$ telle que

$$\Phi(a \cdot b) = \Phi(a) \cdot \Phi(b)$$

$\forall a, b \in \mathbb{G}_1$. On aura ainsi $\Phi(e) = e$ et $\Phi(g^{-1}) = \Phi(g)^{-1}$.

**Définition 2.1.1.** Un anneau est un triplet ($\mathbb{R}$,+,$\cdot$), où + et $\cdot$ sont des opérations binaires et les propriétés suivantes sont satisfaites :

- $\mathbb{R}$ est un groupe Abélien en ce qui concerne les opérations $\cdot$ et + .
- $\cdot$ est associative.

- La loi de distribution est satisfaite : c.-à-d., $\forall a,\ b,\ c \in \mathbb{R}$ on a $a \cdot (b+c) = a \cdot b + a \cdot c$ et $(b+c) \cdot a = b \cdot a + c \cdot a$.

L'anneau est appelé *avec identité* si l'anneau a un élément neutre multiplicatif et *commutatif* si $\cdot$ est commutative.

**Définition 2.1.2.** Soit $\mathbb{G}$ un anneau avec un élément zéro 0. Un *corps* $\mathbb{F}$ est un anneau commutatif avec un élément neutre où chaque élément $g \in \mathbb{G},\ g \neq 0$ est inversible. L'ordre d'un élément $g \in \mathbb{F}$ est le plus petit entier positif $i$ tel que $g^i = 1$, où 1 est l'élément neutre multiplicatif de $\mathbb{F}$. La caractéristique $\chi(\mathbb{F})$ de $\mathbb{F}$ est le plus petit entier positif $i$ telle que l'équation

$$\underbrace{1+1+\cdots+1}_{i} = 0$$

est satisfaite (si $i$ est inexistant, $\chi(\mathbb{F}) = 0$). $\chi(\mathbb{F})$ est soit nulle, soit un nombre premier.

Un exemple de corps fini est $\mathbb{F}_p$ défini par $(\{0, 1, \cdots, p-1\}, +(\mathrm{mod}p), \cdot(\mathrm{mod}p))$, où $p$ est un nombre premier. En plus, on a $\chi(\mathbb{F}_p) = p$.

### 2.1.2 Extension de corps

À présent, nous donnons une définition des extensions de corps ainsi que la représentation des éléments d'une extension de corps.

**Définition 2.1.3.** Soit $\mathbb{F}_p$ un corps fini, avec un nombre premier $p$. Le corps $\mathbb{F}_{p^m}$ avec un entier $m > 1$ est appelé une *extension de corps* du sous-corps $\mathbb{F}_p$.

Pour représenter les éléments d'une extension de corps, on utilise l'anneau de classes de résidus $F[x]/(f)$, où $f$ est un polynôme irréductible sur $\mathbb{F}_p$ avec degré $m$.

On appelle $f$ le polynôme de corps. Cependant, les éléments $A \in \mathbb{F}_{p^m}$ sont représentés par des polynômes dans l'anneau de classes de résidu comme suit :

$$A = a_{m-1}x^{m-1} + \cdots + a_1x^1 + a_0x^0$$

avec $a_i \in \mathbb{F}_p$, $i = m-1, \ldots, 0$. Les opérations arithmétiques sur l'extension de corps sont des opérations sur les polynômes modulo un polynôme de corps (irréductible).

## 2.2 Les courbes elliptiques

Soit $\mathbb{F}$ un corps dont la caractéristique est différente de 2 et 3. Une courbe elliptique $E$ est définie par l'équation (affine) de Weierstrass

$$y^2 = x^3 + ax + b \tag{2.2.1}$$

$$\text{avec } a, b \in \mathbb{F}$$

On suppose que $a^3 + 27b^2 \neq 0$ dans $\mathbb{F}$. De ce fait, une courbe elliptique est l'ensemble de points $(x, y) \in \mathbb{F}^2$ qui satisfait l'équation (2.2.1), plus un point à l'infini noté $\mathcal{O}$.

On dit que $E$ est définie sur le corps $\mathbb{F}$. $E(\mathbb{F})$ désigne l'ensemble de points avec les deux coordonnées sur $\mathbb{F}$ (et sur $E$) avec le point $\mathcal{O}$.

En général, nous allons travailler avec des courbes elliptiques définies sur un corps fini $\mathbb{F} = \mathbb{F}_q$ qui consiste en q éléments, où $q = p^m$ est une puissance d'un nombre premier p. Dans ce cas, la clôture algébrique de $\mathbb{F}$ est donnée par $\bigcup_{i \geq 1} \mathbb{F}_{q^i}$. Ici, $p$ est appelé *la caractéristique* du corps $\mathbb{F}$. Il est généralement supérieur à 3. L'ensemble des

points sur la courbe elliptique avec l'opération addition forme un groupe. Il a comme élément neutre le *point à l'infini* $\mathcal{O}$.

Dans ce qui suit, nous donnons la formule explicite pour l'addition de deux points sur la courbe $E$ définie sur un corps fini de caractéristique $p > 3$.

Soit $P = (x_1, y_1)$ , $Q = (x_2, y_2) \in E\backslash\{\mathcal{O}\}$. Le point $-P$ est donné par $(x_1, -y_1)$. Supposons que $Q \neq -P$, alors $P + Q = (x_3, y_3)$, où

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

et

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & si \quad P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & si \quad P = Q \end{cases}$$

À partir de cette addition sur les courbes elliptiques, nous pouvons également définir une multiplication scalaire. Précisément, soit $m \in \mathbb{Z}$, et $P \in E$, la *multiplication* du point $P$ par $m$ est donnée par

$[m]P = P + P + \cdots + P$ ($m$ termes) pour $m > 0$

$[0]P = \mathcal{O}$

$[-m]P = [m](-P)$ pour $m < 0$.

On définit *l'ordre* d'un point $P \in E$ comme étant le plus petit entier positif $k$ tel que $[k]P = \mathcal{O}$. Si un tel entier est inexistant alors l'ordre est infini. Si $[n]P = \mathcal{O}$ pour $P \in E$ alors $P$ est appelé un point de *n-torsion*. Le sous-groupe de points de n-torsion dans $E$ est dénoté par $E[n]$.

Par conséquent on a

$$E[n] = \{P \in E : [n]P = \mathcal{O}\} \tag{2.2.2}$$

Soit $E$ une courbe elliptique définie sur $\mathbb{F}_q$ où $q = p^m$ une puissance d'un nombre premier. Le nombre de points dans $E(\mathbb{F}_q)$, appelé l'ordre de la courbe elliptique, est désigné par $\#E(\mathbb{F}_q)$. *La trace de Frobenius*, ou tout simplement, la trace d'une courbe est la valeur $t$ qui satisfait $\#E(\mathbb{F}_q) = q+1-t$. La courbe elliptique $E(\mathbb{F}_q)$ est appelée *supersingulière* si la caractéristique $p$ divise $t$, sinon la courbe est non-supersingulière. En d'autres mots, $E(\mathbb{F}_q)$ est supersingulière si $t \equiv 0 \pmod{p}$. Voici le théorème de Hasse.

**Théorème 2.2.1.** *( **[Sil86] , théorème 5.1.1** ) La trace $t$ d'une courbe satisfait* $|t| \leq 2\sqrt{q}$.

Une application de $E$ vers $E$ est appelée un endomorphisme. L'ensemble des endomorphismes sur $E$ forme un groupe désigné par $End(E)$. L'application multiplication par un entier $m$ définie de $E$ vers $E : P \longrightarrow [m]P$ est un endomorphisme. En outre, pour toute courbe elliptique définie sur un corps fini $\mathbb{F}_q$, il existe un autre endomorphisme, connu sous le nom d'*endomorphisme de Frobenius*. L'endomorphisme de Frobenius est défini par $\Phi : P(x,y) \longrightarrow \Phi(P) = (x^q, y^q)$ , avec $\Phi(\mathcal{O}) = \mathcal{O}$. Ainsi, $P \in E(\mathbb{F}_q)$ si et seulement si $\Phi(P) = P$.

## 2.3 Les fonctions sur une courbe elliptique

Soit $\mathbb{K} = \mathbb{F}_q$ un corps fini et soit la courbe $E/\mathbb{K}$ donnée par l'équation de Weierstrass simplifiée (2.2.1). On définit la fonction $r(x, y) \in \mathbb{K}[x, y]$ par :

$$y^2 - x^3 - ax - b \tag{2.3.1}$$

On sait que $r(x, y) = 0$ pour tout point $P(x, y)$ sur la courbe $E$. L'ensemble des fonctions $f \in \mathbb{K}[x, y]/ < E >$ est désigné par $\mathbb{K}[E]$.

Le corps des fractions de $\mathbb{K}[E]$ est désigné par $\mathbb{K}(E)$. Un élément de $\mathbb{K}(E)$ est appelé une *fonction rationnelle*. Une fonction rationnelle non nulle $f \in \mathbb{K}(E)^*$ est dite *définie* sur un point $P \in E\backslash\{\mathcal{O}\}$ si $f$ peut être écrite sous la forme $f = g/h$, pour $g, h \in \mathbb{K}[E]$, avec $h(P) \neq 0$. Dans ce cas, l'évaluation de $f$ sur $P$, donnée par $f(P) = g(P)/h(P)$, est bien définie. Par ailleurs, on dit que $f$ a un zéro sur $P$ si $f(P) = 0$, et un pôle si $f$ est indéfinie sur $P$ (c.-à-d., $f(P) = \infty$). Dans ce qui suit, nous verrons la notion de la multiplicité des zéros et pôles d'une fonction rationnelle $r(x, y) \in \mathbb{K}(E)$ sur un point $P \in E(\mathbb{K})$.

**Théorème 2.3.1.** *( [Yor92]) Pour chaque point $P \in E$, il existe une fonction rationnelle $u \in \mathbb{K}(E)$, telle que $u(P) = 0$ et $u$ possède la propriétés suivantes; toute fonction non nulle $f \in \mathbb{K}(E)$, avec $f(P) = 0$, peut être écrite comme suit*

$$f = u^d s,$$

*Pour un certain entier $d$, et une certaine fonction $s \in \mathbb{K}(E), s(P) \neq 0, \infty$.*

Une fonction $u$ avec les propriétés du théorème précédent (2.3.1) est appelée *une fonction (ou un paramètre) d'uniformisation*. La valeur de $d$ ne dépend pas du choix

de $u$. Le théorème suivant définit une manière pour trouver un paramètre d'uniformisation.

**Théorème 2.3.2.** *( [MEN93] ) Soit $P \in E$. Si $l : ax + by + c = 0$ est une ligne quelconque traversant $P$ et qui n'est pas tangente à $P$, alors $l$ est un paramètre d'uniformisation pour $P$.*

Soit $u$ un paramètre d'uniformisation pour $P \in E$. La fonction non nulle $f \in \mathbb{K}(E)$ peut être décomposée comme suit ; $f = u^d s$, où $s \in \mathbb{K}(E)$, $s(P) \neq 0, \infty$. Alors *l'ordre de f sur P*, dénoté $ord_P(f)$ , égal à $d$.
Si $P$ est un zéro de $f$ alors $ord_P(f) > 0$ et le zéro est dit avoir *multiplicité* $ord_P(f)$ . Dans le cas où $P$ est un pôle, alors $ord_P(f) < 0$ et la multiplicité du pôle est donnée par $-ord_P(f)$. À noter que si P est ni un zéro ni un pôle alors $ord_P(f) = 0$.

## 2.4 Théorie des diviseurs

Les diviseurs sont une partie cruciale des couplages de Weil et Tate. Dans cette section, nous donnons quelques définitions et résultats sur la théorie des diviseurs. Pour de plus amples détails voir [Sil86], [Wen98] et [MEN93].

Un *diviseur* $\mathcal{D}$ est défini comme étant une somme formelle de points d'une courbe elliptique

$$\sum_{P \in E} n_P(P)$$

où $n_P \in \mathbb{Z}$ sont tous nuls sauf un nombre fini. On désigne par $Div(E)$ le groupe de diviseurs d'une courbe $E$, où l'opération addition est donnée par :

$$\sum_{P\in E} n_P(P) \quad + \quad \sum_{P\in E} m_P(P) \quad = \quad \sum_{P\in E} (n_P + m_P)(P)$$

Le *support* d'un diviseur $\mathcal{D} = \sum_{P\in E} n_P(P) \in Div(E)$, est donné par l'ensemble de points :

$$supp(\mathcal{D}) \ = \ \{P \in E \ / \ n_P \neq 0\}$$

Son *degré*, noté $deg(\mathcal{D})$ est défini par :

$$deg(D) \ = \ \sum_{P\in E} n_P$$

Il est facile de vérifier que l'ensemble de diviseurs de degré zéro, muni de la loi d'addition, désigné par $Div^0(E)$, est un sous-groupe du groupe $Div(E)$. A travers ce rapport, nous allons seulement considérer les diviseurs de degré zéro. Puisque le nombre de zéros et pôles d'une fonction rationnelle non nulle $f \in \mathbb{K}(E)^*$ est fini, nous pouvons définir *le diviseur d'une fonction* $f$, noté $div(f)$, comme suit :

$$div(f) \ = \ \sum_{P\in E} ord_P(f)(P)$$

Notons que $div(f) = 0$ si et seulement si $f$ est une fonction constante.

Un diviseur $\mathcal{D} \in Div(E)$ est appelé *principal* si $D = div(f)$ pour une certaine fonction $f$.

En outre, deux diviseurs $\mathcal{D}_1, \mathcal{D}_2 \in Div(E)$ sont *(linéairement) équivalents*, et on écrit $\mathcal{D}_1 \sim D_2$, si $\mathcal{D}_1 - \mathcal{D}_2$ est principal, autrement dit, si on peut trouver une fonction rationnelle $f$ telle que $\mathcal{D}_1 = D_2 + div(f)$.

Lorsqu'une fonction $f \in \mathbb{K}(E)^*$ peut être décomposée sous la forme $f = u^d s$, où $s \in \mathbb{K}(E), s(P) \neq 0, \infty$, l'ordre de $f$ sur $P$, $ord_P(f) = d$.

Le théorème suivant explique les particularités d'un diviseur principal.

**Théorème 2.4.1.** *( [Sil86], corollaire 3.3.5) Soit* $\mathcal{D} = \sum_{P \in E} n_P(P)$ *un diviseur. Alors* $\mathcal{D}$ *est principal si et seulement si*

$$\sum_{P \in E} n_P = 0 \quad et \quad \sum_{P \in E} [n_P]P = \mathcal{O} \tag{2.4.1}$$

Dans ce qui suit, on montre comment évaluer une fonction $f \in \mathbb{K}(E)$ d'un diviseur $\mathcal{D} = \sum_{P \in E} n_P = 0$ qui satisfait la condition[1] $supp(div(f)) \cap supp(\mathcal{D}) = \emptyset$ (on dit ainsi que $\mathcal{D}$ et $div(f)$ ont des supports disjoints). L'évaluation de $f$ dans $\mathcal{D}$ est donnée par

$$f(\mathcal{D}) = \prod_{P \in supp(D)} f(P)^{n_P} \tag{2.4.2}$$

Cette évaluation est bien définie (différente de $0, \infty$) car $\mathcal{D}$ et $div(f)$ ont des supports disjoints.

**Lemme 2.4.2.** *( [Lyn]) Soit* $\mathcal{D} \in Div(E)$. *Alors il existe un point unique* $P \in E$ *tel que* $\mathcal{D} \sim (P) + (deg(D) - 1)(\mathcal{O})$

Il en résulte de ce lemme que si $\mathcal{D}$ est un diviseur de degré 0, alors il est équivalent à $(P) - (\mathcal{O})$ pour un certain point $P$.

## 2.5 Calcul de la fonction d'un diviseur principal

En vertu de la section précédente, pour chaque diviseur de degré zéro $\mathcal{D} \in Div^0(E)$, il existe un point unique $P \in E$ tel que $\mathcal{D} \sim (P) - (\mathcal{O})$. Autrement dit, $\mathcal{D}$ peut être écrit dans ce qu'on appelle la forme canonique

[1] Cette restriction est justifiée par la raison suivante : supposons que $\mathcal{D}$ et $div(f)$ n'ont pas de supports disjoints, alors il existe un point R tel que $f(R) = 0$ ou $f(R) = \infty$. Ainsi, ces points apparaissent sûrement dans la multiplication, et le résultat final est soit 0 soit $\infty$. D'où la contrainte que $\mathcal{D}$ et $div(f)$ doivent avoir des supports disjoints.

$$\mathcal{D} = (P) - (\mathcal{O}) + div(f) \tag{2.5.1}$$

où $f \in \mathbb{K}(E)$.

Pour un diviseur $\mathcal{D}$ de degré zéro, on montre comment calculer $P$ et $f$ [MEN93], [Wen98]. À cette fin, on donne la formule explicite pour additionner deux diviseurs dans la forme canonique. Rappelons que $div(f_1 f_2) = div(f_1) + div(f_2)$ et $div(f_1/f_2) = div(f_1) - div(f_2)$. Soient $\mathcal{D}_1, \mathcal{D}_2 \in Div^0(E)$ donnés par :

$$\mathcal{D}_1 = (P_1) - (\mathcal{O}) + div(f_1)$$
$$\mathcal{D}_2 = (P_2) - (\mathcal{O}) + div(f_2)$$

Soit $P_3 = P_1 + P_2$, et soit $l : u_1 y + u_2 x + u_3 = 0$, l'équation de la ligne qui passe à travers $P_1$ et $P_2$, et $v : x + v_1 = 0$, la ligne verticale qui passe à travers le point $P_3$. Si $P_1 = P_2$ alors $l$ est la ligne tangente à travers $P_1$, et si $P_3 = \mathcal{O}$ (c.-à-d., $P_1$ et $P_2$ sont symétriques par rapport à l'axe des $x$) alors $v = 1$. Alors

$div(l) = (P_1) + (P_2) + (-P_3) - 3(\mathcal{O})$

et

$div(v) = (P_3) + (-P_3) - 2(\mathcal{O})$

La somme des diviseurs $\mathcal{D}_1 + \mathcal{D}_2$ peut être écrite sous la forme

$\mathcal{D}_1 + \mathcal{D}_2 = (P_1) + (P_2) - 2(\mathcal{O}) + div(f_1 f_2)$

$= (P_3) - (\mathcal{O}) + div(l) - div(v) + div(f_1 f_2)$

$= (P_3) - (\mathcal{O}) + div(f_1 f_2 f_3)$

où $f_3 = l/v$. À noter que $f_3$ est définie pour tous les points de $\mathbb{K}(E)$ sauf pour $P_3$ et $-P_3$.

Décrivons une méthode pour écrire le diviseur principal $\mathcal{D} = \sum_{i=1}^{n} a_i(P_i)$ comme un diviseur d'une fonction $f \in \mathbb{K}(E)$. Puisque $deg(\mathcal{D}) = 0$, nous pouvons écrire $\mathcal{D} = \sum_{i=1}^{n} a_i(P_i) = \sum_{i=1}^{n} a_i\ ((P_i) - (\mathcal{O}))$.

Nous introduisons la notion d'une *chaîne d'addition* [Wen98]. Soit $a \in \mathbb{N}$, alors la chaîne d'addition de $a$ est la séquence $1 = d_1, d_2, \ldots, d_t = a$, telle que tout $d_j, 2 \leq j \leq t$, peut être écrit comme $d_j = d_k + d_l$, où $k, l < j$. Par exemple, pour l'entier 13 on a la chaîne suivante ; $d_1 = 1, d_2 = d_1 + d_1 = 2, d_3 = d_2 + d_2 = 4, d_4 = d_3 + d_3 = 8, d_5 = d_3 + d_4 = 12, d_6 = d_5 + d_1 = 13$.

Revenons au diviseur $\mathcal{D}$, soit $d_1^{(i)}, \ldots, d_{t_i}^{(i)}$ une chaîne d'addition pour $a_i$, alors pour chaque $i = 1, \ldots, n$ on utilise la méthode décrite plus haut pour exprimer $d_j^{(i)}((P_i) - (\mathcal{O}))$ pour $j = 1, \ldots, t_i$ dans la forme canonique.

Soit $(P_i') - (\mathcal{O}) + div(f_i)$ la forme canonique de $a_i((P_i) - (\mathcal{O}))$. Le résultat final est obtenu en faisant l'addition des diviseurs $(P_i') - (\mathcal{O}) + div(f_i)$ pour $i = 1, \ldots, n$.

### 2.5.1 Exemple

Considérons la courbe elliptique $E/\mathbb{F}_{11} : y^2 = x^3 + x$ . Le *tableau 2.1* montre les points sur $E(\mathbb{F}_{11})$. Le nombre de points $\#E(F_{11}) = 12(= p + 1)$, nous avons $t = q + 1 - \#E(F11) = 0$, ainsi $E$ est supersingulière.

Il est clair que le diviseur $\mathcal{D} = 6(P_8) - 6(\mathcal{O})$ est principal, car l'ordre de $P_8$ est 6, une chaîne d'addition de 6 est donnée par 1, 2, 4, 6.

| Point | Ordre | Point | Ordre |
|---|---|---|---|
| $P_0 = \mathcal{O}$ | 1 | $P_6 = (8,5)$ | 12 |
| $P_1 = (0,0)$ | 2 | $P_7 = (8,6)$ | 12 |
| $P_2 = (5,3)$ | 3 | $P_8 = (9,1)$ | 6 |
| $P_3 = (5,8)$ | 3 | $P_9 = (9,10)$ | 6 |
| $P_4 = (7,3)$ | 12 | $P_{10} = (10,3)$ | 4 |
| $P_5 = (7,8)$ | 12 | $P_{11} = (10,8)$ | 4 |

TAB. 2.1 – Les points de la courbe $y^2 = x^3 + x$

Ainsi, la fonction rationnelle $f$ avec $div(f) = \mathcal{D}$ est calculée comme suit :

$$2(P_8) - 2(\mathcal{O}) = ((P_8) - (\mathcal{O})) + ((P_8) - (\mathcal{O})) = (P2) - (\mathcal{O}) + div\left(\tfrac{y+10x+8}{x+6}\right)$$

$$4(P_8) - 4(\mathcal{O}) = (2(P_8) - 2(\mathcal{O})) + (2(P_8) - 2(\mathcal{O}))$$

$$= (P_3) - (\mathcal{O}) + div\left(\tfrac{(y+10x+8)^2}{(x+6)^2} \cdot \tfrac{y+2x+9}{x+6}\right)$$

$$6(P_8) - 6(\mathcal{O}) = (2(P_8) - 2(\mathcal{O})) + (4(P_8) - 4(\mathcal{O}))$$

$$= div\left(\tfrac{(y+10x+8)^3}{(x+6)^3} \cdot \tfrac{y+2x+9}{x+6} \cdot \tfrac{(x+6)}{1}\right)$$

La fonction rationnelle $f$ est donnée par

$$f = \tfrac{(y+10x+8)^3 \cdot (y+2x+9)}{(x+6)^3}$$

## 2.6 Les couplages bilinéaires sur les courbes elliptiques

### 2.6.1 Le couplage de Weil

Soit $E$ une courbe elliptique définie sur $\mathbb{F}_q$. Soient $m$ un entier positif relativement premier à $q$ et $\mu_m$ le groupe des $m^{\text{ième}}$ racine de l'unité tel que $m \mid (q^k - 1)$.
Soient $P, Q \in E[m]$.
Soient $\mathcal{D}_1, \mathcal{D}_2 \in Div^0(E)$ deux diviseurs de degré zéro tels que $\mathcal{D}_1 \sim (P) - (\mathcal{O})$, $\mathcal{D}_2 \sim (Q) - (\mathcal{O})$ et $supp(\mathcal{D}_1) \cap supp(\mathcal{D}_2) = \emptyset$.
Par conséquent, d'après le théorème (2.3.2), $m\mathcal{D}_1$ et $m\mathcal{D}_2$ sont des diviseurs principaux. Soient $f_{\mathcal{D}_1}$ et $f_{\mathcal{D}_2} \in \mathbb{F}_q(E)$ deux fonctions telles que $div(f_{\mathcal{D}1}) = m\mathcal{D}_1$ et $div(f_{\mathcal{D}_2}) = m\mathcal{D}_2$.

**Définition 2.6.1.** Le couplage de Weil est la fonction $e_m : E[m] \times E[m] \longrightarrow \mu_m$ défini par $e_m(P, Q) = f_{D_1}(D_2)/f_{D_2}(D_1)$.

Le couplage de Weil satisfait les propriétés suivantes ( [Sil86] pages 96-98) :

1. *Bien défini* : La valeur de $e_m(P, Q)$ est indépendante des choix de $\mathcal{D}_1$, $\mathcal{D}_2$, $f_{\mathcal{D}_1}$, $f_{D_2}$.
2. *Identité* : Pour tout $P \in E[m]$, $e_m(P, P) = 1$.
3. *Non dégénéré* : Si $e_m(P, Q) = 1$ pour tout $P \in E[m]$ alors $Q = \mathcal{O}$.
4. *Bilinéarité* : Pour tout $P, Q, R \in E[m]$, $e_m(P + Q, R) = e_m(P, R)e_m(Q, R)$ et $e_m(P, Q + R) = e_m(P, Q)e_m(P, R)$.
5. *Alternation* : Pour tout $P, Q \in E[m]$, $e_m(P, Q) = e_m(Q, P)^{-1}$.

### 2.6.2 Le couplage de Tate

Soit $E$ une courbe elliptique définie sur $\mathbb{F}_q$. Soit $m$ un entier positif relativement premier à $q$.

Soit $k$ un entier positif tel que :

$$m \mid (q^k - 1) \quad et \quad m \nmid (q^s - 1) \quad pour \quad 0 < s < k \tag{2.6.1}$$

Soient $P, Q \in E[m]$.

Soient $\mathcal{D}_1, \mathcal{D}_2 \in Div^0(E)$ deux diviseurs de degré zéro tels que $\mathcal{D}_1 \sim (P) - (\mathcal{O})$, $\mathcal{D}_2 \sim (Q) - (\mathcal{O})$ et $supp(\mathcal{D}_1) \cap supp(\mathcal{D}_2) = \emptyset$.

Par conséquent, d'après le théorème (2.3.2), $m\mathcal{D}_1$ est un diviseur principal. Soit $f_{\mathcal{D}_1} \in \mathbb{F}_q(E)$ telle que $div(f_{D_1}) = m\mathcal{D}_1$.

**Définition 2.6.2.** Le couplage de Tate est la fonction $t$ : $E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \longrightarrow \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^m$ défini par $t(P, Q) = f_{\mathcal{D}_1}(\mathcal{D}_2)$.

Le quotient de groupes $E/mE$ peut être considéré comme étant l'ensemble des classes d'équivalences de $E$ sous la relation d'équivalence $P \equiv Q$ si et seulement s'il existe $R \in E$ tel que $P = Q + mR$.

Une interprétation similaire peut être attribuée au $F^*_{q^k}/(F^*_{q^k})^m$. Le couplage de Tate satisfait les propriétés suivantes (voir [Sil86] pour les preuves) :

1. *Bien défini* : $t(\mathcal{O}, Q) = 1$ (l'unité dans $\mathbb{F}_{q^k}$ pout tout $Q \in E(\mathbb{F}_{q^k})$, et $t(P, Q) \in (\mathbb{F}_{q^k})^m$ pour tout $P \in E(\mathbb{F}_{q^k})[m]$ et $Q \in E(F_{q^k})/mE(F_{q^k})$.
   En outre, $t(P, Q)$ est indépendante des choix de $\mathcal{D}_1$, $\mathcal{D}_2$, $f_{\mathcal{D}_1}$.
2. *Non dégénéré* : Si $t(P, Q) = 1$ pour tout $P \in E[m]$ alors $Q = \mathcal{O}$.
   Inversement, pour tout $P \neq \mathcal{O}$, $\exists Q \in E(F_{q^k})[m]$ tel que $t(P, Q) \neq 1$.

3. *Bilinéarité :* Pour tout $P, Q, R \in E[m], t(P+Q, R) = t(P, R)t(Q, R)$ et $t(P, Q+R) = t(P, Q)t(P, R)$.

Notons que $t(P, Q)^m = t([m]P, Q) = t(\mathcal{O}, Q) = 1 \in \mathbb{F}_{q^k}$, cependant, $t(P, Q)$ est la $m^{\text{ième}}$ racine de l'unité. Enfin, lorsque $m^2 \mid \#E(\mathbb{F}_{q^k})$, le couplage de Weil et celui de Tate sont liés par l'équation suivante : $e_m(P, Q) = t(P, Q)/t(Q, P)$ pour $P, Q \in E[m]$ si $P, Q$ sont indépendants.

## 2.7 Conclusion

Dans ce chapitre, nous avons montré comment construire une fonction rationnelle à partir d'un diviseur principal de degré zéro. Cette information est importante puisqu'elle nous permettra de comprendre le fonctionnement de l'algorithme de Miller qui n'est, en fait, qu'une méthode itérative, efficace et pratique pour évaluer une telle fonction sur un point donné de la courbe elliptique. Nous avons également défini les couplages bilinéaires de Weil et de Tate, dont le calcul fait appel à l'algorithme de Miller. Ces couplages, particulièrement celui de Tate, représentent la pierre angulaire du système de chiffrement projeté.

# Chapitre 3

# Calcul des couplages de Weil et de Tate

Le calcul du couplage de Weil/Tate est un processus onéreux qui fait appel à des opérations calculatoires complexes. Au départ, lorsque les couplages ont été proposés, le meilleur algorithme connu était exponentiel dans la taille des entrées. Miller propose dans [Mil86] un algorithme efficace qui est linéaire dans la taille des entrées. Le but ultime de l'algorithme de Miller est de construire une fonction dont on connaît le diviseur principal. Le reste de ce chapitre est subdivisé comme suit :
La section 3.1 est consacrée à la présentation du principe de fonctionnement de l'algorithme de Miller. Le calcul du couplage de Weil et celui de Tate sont présentés respectivement dans les sections 3.2 et 3.3. On traite également dans la section 3.3 de la relation entre les deux couplages en dressant une comparaison.
La section 3.4 comporte des améliorations au couplage de Tate en agissant à la fois sur l'algorithme de Miller, sur le choix du groupe des entrées et sur l'algorithme de calcul. On termine cette section et le chapitre par un exemple de calcul du couplage de Tate sur la courbe supersingulière usuelle $E_{1,0}$ définie sur le corps fini $\mathbb{F}_p$ avec $p = 43$.

## 3.1 L'algorithme de Miller

Soit $E$ une courbe elliptique définie sur $\mathbb{F}_q$. Le résultat suivant est utilisé à plusieurs reprises pour calculer les couplages de Weil et de Tate.

On définit $g_{u,v}$ : $E \longrightarrow \mathbb{F}_q$ comme étant la ligne passant par les points $u$ et $v$ [1].

Si $u = v$ alors $g_{u,v}$ est la ligne tangente de $E$ à $u$, et si $u$ ou $v$ est le point à l'infini $\mathcal{O}$, $g_{u,v}$ est la ligne verticale à travers l'autre point.

On écrit $g_u$ comme une abréviation de $g_{u,-u}$. Dans le cas des coordonnées affines, pour $u = (x_u, y_u)$, $v = (x_v, y_v)$ et $Q = (x, y)$, on a :

$g_{u,v}(\mathcal{O}) = 1$

$g_{u,u}(Q) = \lambda_1(x - x_u) + y_u - y, \quad Q \neq \mathcal{O}$

$g_{u,v}(Q) = \lambda_2(x - x_u) + y_u - y, \quad u \neq v$

$g_u(Q) = x - x_u, \quad Q \neq \mathcal{O}$

où $\lambda_1 = \dfrac{3x_u^2 + a}{2y_u}$, et $\lambda_2 = \dfrac{y_v - y_u}{x_v - x_u}$

**Lemme 3.1.1.** *(**lemme 6 dans [BLS03]**) Soit $P$ un point sur $\mathbb{F}_{q^k}$ et $f_c$ une fonction avec le diviseur $div(f_c) = c(P) - (cP) - (c-1)\mathcal{O}$, $c \in \mathbb{Z}$. Pour tout $a, b \in \mathbb{Z}$,*

$$f_{a+b}(Q) = f_a(Q) \cdot f_b(Q) \cdot g_{aP,bP}(Q)/g_{(a+b)P}(Q). \tag{3.1.1}$$

À noter que $div(f_0) = div(f_1) = 0$, ainsi $f_0(Q) = f_1(Q) = 1$. En outre, $f_{a+1}(Q) = f_a(Q) \cdot g_{aP,P}(Q)/g_{(a+1)P}(Q)$ et $f_{2a}(Q) = f_a(Q)^2 \cdot g_{aP,aP}(Q)/g_{2aP}(Q)$. Rappelons que $m \geq 0$ est l'ordre de $P$. Soit $m = \{m_t, \cdots, m_1, m_0\}$ sa représentation binaire où $m_i \in \{0, 1\}$ et $m_t \neq 0$.

L'algorithme de Miller calcule $f(Q) = f_m(Q)$, $Q \neq \mathcal{O}$, en utilisant conjointement les formules précédentes avec la méthode de *doubler et additionner* pour calculer $[m]P$.

[1] si la ligne à travers $u$ et $v$ a pour équation $ax + by + c = 0$, alors $g_{u,v} = ax + by + c$.

Nous pouvons calculer successivement $f_{(m_t,\cdots,m_i)_2}$ pour $i = t-1, \cdots, 0$ par doubler et additionner $f_{(m_t,\cdots,m_{i+1})_2}$ si $m_i = 1$, et seulement doubler si $m_i = 0$. Le résultat final est $g = f_m = f_{(m_t,\cdots,m_1,m_0)_2}$, où $div(g) = m(P) - m(\mathcal{O})$ comme prévu.
La fonction $g$ a besoin d'être évaluée sur un diviseur $\mathcal{D}$ équivalent à $(Q) - (\mathcal{O})$.
Prenons par par exemple $\mathcal{D} = (Q + Q') - (Q')$ pour $Q' \in E(F_{q^k})$, on a besoin de calculer $g(Q + Q')/g(Q')$. Au lieu de construire tout d'abord $g$ complètement puis l'évaluer sur $\mathcal{D}$, il est plus efficace de faire l'évaluation au fur et à mesure, c'est-à-dire garder trace des évaluations de toutes les fonctions intermédiaires $f_{(m_t,\cdots,m_i)_2}$ au lieu des fonctions elles-mêmes. Cela évite de gaspiller excessivement de l'espace de stockage [Wen98].

**Algorithme 1** L'algorithme de Miller

**Entrées:** les points $P \in E(\mathbb{F}_{q^k})[m]$, $Q \in E(\mathbb{F}_{q^k})$ et un entier $m$ avec sa représentation binaire $m = \sum_{i=0}^{t} b_i 2^i$.
**Sorties:** $f_P(m\mathcal{D}) \in \mathbb{F}_{q^k}$.

```
f := 1   V := P
Pour i = t − 1, t − 2, · · · , 1, 0 Faire
  f := f^2 ġ_{V,V}(Q)/g_{2V}(Q)
  V := 2V
  Si m_i = 1 Alors
    f := f · g_{V,P}(Q)/g_{V+P}(Q)
    V := V + P
  Fin Si
Fin Pour
retourner f
```

Les prochaines sections montrent d'autres simplifications de cet algorithme afin de gagner en temps d'exécution.

## 3.2 Calcul du couplage de Weil

Soit $E$ une courbe elliptique définie sur $\mathbb{F}_q$, soit $m$ un entier positif avec $gcd(m, q) = 1$ et soient $P, Q, \in E[m]$.

Nous montrons comment calculer le couplage de Weil $e_m(P, Q)$ pour $P, Q \neq \mathcal{O}$ (si $P$ ou $Q = \mathcal{O}$, $e_m(P, Q) = 1$).

Choisir aléatoirement $T, U \in E\backslash\{\mathcal{O}\}$ tel que $P+T \neq U, U+Q$ et $T \neq U, U+Q$. Soit $\mathcal{D}_1 = (P+T)-(T)$ et $\mathcal{D}_2 = (Q+U)-(U)$. Alors, $\mathcal{D}_1 \sim (P)-(\mathcal{O})$ et $\mathcal{D}_2 \sim (Q)-(\mathcal{O})$. On veut trouver deux fonctions non-nulles $f_{\mathcal{D}_1}$ et $f_{\mathcal{D}_2} \in \mathbb{F}_q(E)$ telles que $div(f_{\mathcal{D}_1}) = m\mathcal{D}_1$ et $div(f_{\mathcal{D}_2}) = m\mathcal{D}_2$ et calculer $f_{\mathcal{D}_1}(\mathcal{D}_2)/f_{\mathcal{D}_2}(\mathcal{D}_1)$. Notons que $m(P+T)-m(T) = div(f_{P+T}) - div(f_T)$. Ainsi,

$$e_m(P, Q) = f_{\mathcal{D}_1}(\mathcal{D}_2)/f_{\mathcal{D}_2}(\mathcal{D}_1) = \frac{f_{P+T}(\mathcal{D}_2)\, f_U(\mathcal{D}_1)}{f_T(\mathcal{D}_2)\, f_{Q+U}(\mathcal{D}_1)} \tag{3.2.1}$$

On utilise l'algorithme de Miller pour calculer $f_{P+T}(Q+U)$, $f_T(Q+U)$, $f_{P+T}(U)$, $f_T(U)$, $f_{Q+U}(P+T)$, $f_{Q+U}(T)$, $f_U(P+T)$, et $f_U(T)$. Il en résulte

$$e_m(P, Q) = \frac{f_{P+T}(Q+U) \cdot f_T(U) \cdot f_{Q+U}(T) \cdot f_U(P+T)}{f_T(Q+U) \cdot f_{P+T}(U) \cdot f_{Q+U}(P+T) \cdot f_U(T)} \tag{3.2.2}$$

Notons que $f_{P+T}(Q+U)$ et $f_{P+T}(U)$ peuvent être calculées en une seule invocation de l'algorithme de Miller, de même pour $f_T(Q+U)$ et $f_T(U)$ , $f_{Q+U}(T)$ et $f_{Q+U}(P+T)$, $f_U(P+T)$ et $f_U(T)$. Ainsi, nous avons besoin de quatre invocations de l'algorithme de Miller pour calculer $e_m(P, Q)$.

## 3.3 Calcul du couplage de Tate

Le calcul du couplage de Tate est beaucoup plus simple comparativement à celui de Weil. Soit $E$ une courbe elliptique définie sur $\mathbb{F}_q$ et soit $m$ un entier positif relativement premier à $q$. Soit $k$ un entier positif tel que $m \mid (q^k - 1)$ et soit $P \in E(\mathbb{F}_{q^k})[m]$ et $Q \in E(\mathbb{F}_{q^k})$. Montrons comment calculer le couplage de Tate $t(P, Q)$ pour $P, Q \neq \mathcal{O}$ (si $P$ ou $Q$ est $\mathcal{O}$, $t(P, Q) = 1$).

Choisir aléatoirement un point $U \in E(\mathbb{F}_{q^k}) \setminus \{\mathcal{O}\}$ tel que $P \neq U, U + Q$ et $U \neq -Q$. Soit $\mathcal{D}_1 = (P) - (\mathcal{O})$ et $\mathcal{D}_2 = (Q + U) - (U) \sim (Q) - (\mathcal{O})$.

Notons que puisque $[m]P = \mathcal{O}$, $div(f_P) = m(P) - (mP) - (m-1)(\mathcal{O}) = m(P) - m(\mathcal{O}) = m\mathcal{D}_1$. Ainsi,

$$t(P, Q) = f_P(\mathcal{D}_2) = \frac{f_P(Q + U)}{f_P(U)} \tag{3.3.1}$$

Cette valeur peut être calculée en une seule invocation de l'algorithme de Miller.

### 3.3.1 Comparaison avec le couplage de Weil

Le but de cette comparaison est de justifier l'utilisation du couplage de Tate qui est beaucoup plus rapide que celui de Weil, et, de ce fait, son usage est de plus en plus utilisé dans les systèmes de chiffrement.

Algébriquement, cette comparaison a seulement un sens lorsque $Q \in E(F_{q^k})[m]$, pour qu'on puisse calculer $t(Q, P)$. Sous cette condition, l'équation (3.2.1) peut s'écrire sous la forme

$$e_m(P, Q) = \frac{f_{\mathcal{D}_1}(P + T)\, f_{\mathcal{D}_2}(U)}{f_{\mathcal{D}1}(T)\, f_{\mathcal{D}2}(Q + U)} = \frac{t(P, Q)}{t(Q, P)} \tag{3.3.2}$$

Visiblement, ce résultat montre que le temps de calcul du couplage de Weil est au moins le double de celui du couplage de Tate.
Subséquemment, le couplage de Tate est beaucoup plus intéressant du point de vue calculatoire. Pour cette raison, nous adoptons le couplage de Tate pour tout le reste de ce mémoire.

## 3.4 Amélioration du couplage de Tate

Dans cette section, nous tentons de présenter les améliorations qui peuvent être apportées au couplage de Tate pour le rendre plus rapide à calculer. La plupart de cette théorie est tirée de [BLS03] et [BKLS02]. Ces améliorations consistent tout d'abord à élever la valeur du couplage à la puissance de $(q^k - 1)/m$ afin d'avoir une valeur unique au lieu d'une valeur d'équivalence (voir le chapitre précédent). Ce type de couplage est appelé *le couplage de Tate réduit.* Par la suite, choisir un des paramètres du couplage (le point $P$) dans le corps de base $\mathbb{F}_q$ au lieu de l'extension de corps $\mathbb{F}_{q^k}$, ce qui aurait pour conséquence l'évaluation de la fonction du couplage sur un point $Q$ au lieu des deux points $Q + U$ et $U$.
En outre, tous les facteurs appartenant au corps de base $\mathbb{F}_q$ seront éliminés, car $\forall a \in \mathbb{F}_q$, $a^{(q^k-1)/m} = 1$.

### 3.4.1 Le couplage de Tate réduit

Rappelons que le résultat du couplage de Tate n'est pas une valeur unique, mais une classe d'équivalence dans $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^m$.
Ainsi, deux éléments $a, b \in \mathbb{F}^*_{q^k}$ sont *équivalents* $(a \equiv b)$ si et seulement s'il existe un $c \in F^*_{q^k}$ tel que $a = bc^m$. Si un seul résultat doit être retourné, comme c'est le cas

dans les applications cryptographyques, on a besoin d'éliminer la $m^{\text{ième}}$ puissance. Ceci est effectué en élevant la valeur du résultat du couplage de Tate à la puissance $(q^k-1)/m$, car $c^{q^k-1}=1$ pour [2] tout $c \in \mathbb{F}_{q^k}^*$.

Par conséquent, la valeur obtenue après cette exponentiation est une $m^{\text{ième}}$ racine de l'unité notée $\mu_m$. Ceci conduit à une définition alternative du couplage de Tate connue sous l'appellation du *couplage de Tate réduit*, en se servant des données de la section (1.6.2) du chapitre précédent, $t_m$ : $E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \longrightarrow \mu_m$ défini par :

$$t_m(P,Q) = f_{\mathcal{D}_1}(\mathcal{D}_2)^{(q^k-1)/m}. \tag{3.4.1}$$

### 3.4.2 Elimination des facteurs non-pertinents

Nous avons vu dans la section précédente que le paramètre de sécurité [3] $k$ satisfait les conditions $m \mid (q^k-1)$ et $m \nmid (q^s-1)$ pour $0<s<k$. Le lemme suivant permet, en faisant usage du théorème de Fermat, l'élimination des facteurs du corps $\mathbb{F}_q$.

**Lemme 3.4.1.** *(Barreto et al. [BLS03]) $q-1$ est un facteur de $(q^k-1)/m$*

*Démonstration.* Considérons la factorisation de $q^k-1=(q-1)\sum_{i=1}^{k-1} q^i$. Puisque le paramètre de sécurité $k>1$ satisfait $m \mid (q^k-1)$ et $m \nmid (q^s-1)$ pour $0<s<k$, d'où $m \mid \sum_{i=1}^{k-1} q^i$, et $q-1$ demeure un facteur de $q^k-1$. □

À partir du lemme 3.4.1, on en déduit que pour tout $a \in \mathbb{F}_{q^k}$, $a^{(q^k-1)}=(a^{q-1})^{(\frac{q^k-1}{q-1})}=1$.

[2] Ceci est expliqué pleinement dans la prochaine section

[3] Dans la littérature, plusieurs noms sont attribués au paramètre $k$. On trouve *embeeding degree*, *security parameter*, *MOV degree*, *security multiplier* et *extension degree*.

Lorsque l'argument du couplage $P$ est restreint au corps de base $\mathbb{F}_q$, par opposition à avoir les deux points $P$ et $Q$ dans l'extension de corps $\mathbb{F}_{q^k}$, l'évaluation de la fonction du couplage $f_{\mathcal{D}_1}$ pourrait être restreinte au point $Q$ au lieu des deux point $(Q+U)$ et $U$ comme évoqué dans la section (2.3). Ce résultat a été proposé initialement par [BKLS02] pour les courbes supersingulières et puis généralisé par la suite dans [BLS03].

**Théorème 3.4.2.** *( [BLS03]) Soit $P \in E(\mathbb{F}_q)[m]$ et $Q \in E(\mathbb{F}_{q^k})$ deux points linéairement indépendants. Alors $t_m(P,Q) = f_{\mathcal{D}_1}(Q)^{(q^k-1)/m}$.*

*Démonstration.* voir [BLS03]. □

Le corollaire suivant permet de se débarrasser des facteurs non-pertinents dans l'algorithme de Miller.

**Corollaire 3.4.3.** *On peut multiplier $f_{\mathcal{D}_1}(Q)$ par tout $x \in \mathbb{F}_q$ sans affecter la valeur du couplage.*

*Démonstration.* Pour calculer le couplage, on doit élever $f_{\mathcal{D}_1}(Q)$ à la puissance $(q^k-1)/m$. À partir du lemme 3.4.1, cette puissance contient un facteur $(q-1)$, d'où $x^{(q^k-1)/m} = 1$. □

Le théorème suivant introduit des simplifications à l'algorithme de Miller en éliminant les dénominateurs.

**Théorème 3.4.4.** *( [BLS03]) Soit $P \in E(\mathbb{F}_q)[m]$. Supposons que $Q = (X,Y) \in E(\mathbb{F}_{q^k})$ et $X \in \mathbb{F}_q$. Alors les dénominateurs $g_{2V}$ et $g_{V+P}$ dans l'algorithme de Miller peuvent être éliminés sans changer la valeur de $t_m(P,Q)$.*

*Démonstration.* Dans l'algorithme de Miller, Les dénominateurs ont la forme $g_U(Q) = x-u$ ( la ligne verticale passant à travers le point $U$) où $x \in \mathbb{F}_q$ est l'abscisse de $Q$ et $u$ est l'abscisse de $U$. D'où, $g_U(Q) \in \mathbb{F}_q$. Par conséquent, à partir du corrolaire 3.4.3, ils peuvent être éliminés sans changer la valeur du couplage. □

### 3.4.3 Modification du couplage de Tate afin d'obtenir une application bilinéaire

Jusque-là, nous avons supposé que le couplage de Tate satisfait la propriété 2 évoquée dans la section 1.6.2, à savoir la propriété en lien avec la non-dégénérescence du couplage. Or, ceci n'est pas systématiquement vrai. Lorsque $P$ et $Q$ appartiennent au même sous-groupe $E[m]$ on a bel et bien $t_m(P,Q) = 1$, car les deux points sont toujours dépendants par une relation du genre $Q = [x]P$ pour un certain $x \in \mathbb{Z}_m$.

Nous montrons une façon de modifier le couplage de Tate afin d'obtenir une application bilinéaire non-dégénérée. Ainsi, lorsque $P$ et $Q \in E[m]$ sont linéairement indépendants d'ordre $m$, alors $t_m(P,Q)$ est une $m^{\text{ième}}$ racine de l'unité. Ceci nous conduit à introduire la notion d'application de distorsion *distorsion map* [JN01], [Ver01] dans la définition suivante.

**Définition 3.4.1.** Soit $m$ un entier relativement premier avec $q$, $E/\mathbb{F}_q$ une courbe elliptique et $P \in E(\mathbb{F}_q)[m]$. Une *application de distorsion* par rapport à $P$ est un endomorphisme $\phi \in End(E)$ qui associe au point $P$ le point $\phi(P)$ qui est linéairement indépendant de $P$.

Barreto et al. [BLS03] proposent une méthode pour construire de telles applications. Joux et al. [JN01] proposent des applications de distorsion pour certaines courbes elliptiques supersingulières récapitulées dans le tableau 3.1.

Lorsqu'une application de distorsion existe, ce qui est le cas pour la plupart des courbes supersingulières [Ver01], on peut modifier le couplage de Tate de telle façon qu'il satisfait la propriété de non-dégénérescence. En voici la définition du couplage

| Corps | Courbe | Endomorphisme | Conditions | Ordre du groupe |
|---|---|---|---|---|
| $\mathbb{F}_p$ | $y^2 = x^3 + ax$ | $(x,y) \longrightarrow (-x, iy)$<br>$i^2 = -1$ | $p \equiv 3 \pmod 4$ | $p+1$ |
| $\mathbb{F}_p$ | $y^2 = x^3 + x$ | $(x,y) \longrightarrow (\xi x, y)$<br>$\xi^3 = 1$ | $p \equiv 2 \pmod 3$ | $p+1$ |
| $\mathbb{F}_{p^2}$ | $y^2 = x^3 + a$ | $(x,y) \longrightarrow (\omega \frac{x^p}{r^{(2p-1)/3}}, \frac{y^p}{r^{p-1}})$<br>$r^2 = a, \quad r \in \mathbb{F}_{p^2}$<br>$\omega^3 = r, \quad \omega \in \mathbb{F}_{p^6}$ | $p \equiv 2 \pmod 3$ | $p^2 - p + 1$ |

TAB. 3.1 – Applications de distorsion pour certaines courbes supersingulières

de Tate modifié.

**Définition 3.4.2.** Soit $m$ un premier et $E/\mathbb{F}_q$ une courbe supersingulière sur laquelle une application de distorsion $\phi$ existe par rapport aux points dans $E(\mathbb{F}q)[m]$ autre que $\mathcal{O}$.
Soit $P \in E(\mathbb{F}_q)[m]$ et $Q \in E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$.
Alors le couplage de Tate modifié de $P$ et $Q$ est défini par $\hat{t}_m$ : $E(\mathbb{F}_{q^k})[m] \times E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k}) \longrightarrow \mu_m$ défini par

$$\hat{t}_m(P,Q) = t_m(P, \phi(Q)) \tag{3.4.2}$$

### 3.4.4 Le couplage de Tate et la courbe $E_{1,0}$

Nous avons mentionné précédemment que les courbes supersingulières présentent un intérêt remarquable pour les couplages bilinéaires. Ceci est particulièrement dû à l'existence d'une application de distorsion sur de telles courbes. Nous étudions présentement la courbe supersingulière $E_{1,0}$[4] : $y^2 = x^3 + x$ définie sur le corps fini $\mathbb{F}_p$ avec $p \equiv 3 \pmod 4$. On définit l'application de distorsion $\phi$ $E/\mathbb{F}_p \longrightarrow E/\mathbb{F}_{p^2}$ : [5] $\phi(P) = \phi(x,y) = (-x, iy)$ avec $i^2 = -1$. Cette application de distorsion permet

[4] Cette courbe fait l'objet de notre étude pour implanter le cryptosystème basé sur l'identité

[5] Les éléments du corps d'extension $\mathbb{F}_{p^2}$ sont représentés sous la forme $a + ib$ où $a, b \in \mathbb{F}_p$, et $i^2 = -1$.

d'obtenir la propriété de non-dégénérescence pour le couplage de Tate $t$ : $t(P, \phi(P)) \neq 1$ pour tout $P$, autre que $\mathcal{O}$ et $P_1(0,0)$, car $\phi(P_1) = P_1$.

#### 3.4.4.1 Simplification du couplage

L'ordre de la courbe $E_{1,0}$ est $n = p + 1$. Soit $m$ un entier relativement premier avec $p$ tel que $m \mid p + 1$. Considérons un élément du corps d'extension $t \in \mathbb{F}_{p^2}$ avec $t = u + iv$ où $u, v \in \mathbb{F}_p$ et $i$ satisfait $i^2 + 1 = 0$ (ou $i^2 = -1$). L'exposant du couplage de Tate dans l'équation (3.4.1) est $z = (p^2 - 1)/m = ((p + 1)/m) \cdot (p - 1)$. Pour calculer $s = \sigma^z \bmod p$, on calcule tout d'abord $t = \sigma^{(p+1)/m} \equiv u + iv$ puis on met

$$\begin{aligned} s = (u + iv)^{p-1} &= (u + iv)^p/(u + iv) \\ &= (u - v)/(u + iv). \end{aligned}$$

En utilisant le fait que $(X+Y)^p \equiv X^p+Y^p \bmod p$ et $i^p = -i$ pour $p \equiv 3 \pmod 4$.

### 3.4.5 Exemple

Dans cette section, nous présentons un exemple de calcul du couplage de Tate accompagné des résultats intermédiaires à chaque étape de l'algorithme de Miller. On utilise la courbe supersingulière $E_{1,0} : y^2 = x^3 + x$ définie sur le corps fini $\mathbb{F}_p$ avec le nombre premier $p = 43$. L'ordre de cette courbe est $\#E(\mathbb{F}_p) = p + 1 = 44$, avec le paramètre de sécurité $k = 2$ et $m = 11$.

Comme évoqué précédemment, pour ce genre de courbe, on peut utiliser l'application de distorsion $\phi(x, y) = (-x, iy)$ avec $i^2 = -1 \in \mathbb{F}_{p^2}$ pour envoyer les points de $m$-torsion sur $E(\mathbb{F}_p)$ aux points de $m$-torsion sur $E(\mathbb{F}_{p^2})$.

Nous utilisons les points de 11-torsion $P = (x_P, y_P) = (31, 18) \in E(\mathbb{F}_p)$ et $Q = \phi(P) = (x_Q, y_Q) = (12, 18i) \in E(\mathbb{F}_{p^2})$, calculons les couplages de Tate $t(P, Q)$ et

$t([3]P, Q)$ séparément, ensuite décidons si $t([3]P, Q) = t(P, Q)^3$.

Pour la représentation binaire de $m = 11 = 1011_2$, nous utilisons l'approche itérative de l'algorithme de Miller, telle qu'illustrée dans l'algorithme (1), avec les valeurs intermédiaires de $V$, $f$ et $f^2$. Pour chaque élément binaire $m_i$ de $m$, l'opération courante est soit doubler et additionner (DBL suivie de ADD) si $m_i = 1$, soit seulement additionner si $m_i = 0$ (l'opération ADD).

Pour la dernière étape, il s'agit de multiplier la valeur intermédiaire du couplage avec $g_{-P,P}(Q)$ qui est égale à 1, ce qui fait que la valeur finale du couplage reste inchangée.

Notons que les dénominateurs $g_{2V}(Q)$ et $g_{V+P}(Q)$ font partie du corps $\mathbb{F}_p$, et de ce fait, sont éliminés de l'algorithme de Miller en vertu du théorème 3.4.4.

Pour obtenir la valeur finale du couplage de Tate, rappelons le résultat obtenu dans la section précédente pour simplifier l'exposant $(p^2-1)/m$ en procédant en deux étapes :

- La première consiste à calculer le résultat intermédiaire $s = f^{(p+1)/m} = f^4 = u + iv$
- La deuxième étape calcule la valeur du couplage en effectuant l'opération $t = s^{p-1} = (u - v)/(u + iv)$.

#### 3.4.5.1 Calcul de $t(P, Q)$

Pour illustrer l'évolution du calcul du couplage $t(P, Q)$, avec $P = (31, 18)$ et $Q = (12, 18i)$, présentons succinctement les valeurs $V$, $f$ et $f^2$ pour chaque étape de l'algorithme de Miller. Il convient, au préalable, de donner les multiples du point $P$ auxquels on a besoin durant le calcul, en l'occurrence, $[2]P = (4, 38)$, $[4]P = (13, 19)$, $[5]P = (14, 36)$, $[10]P = (31, 25)$ et $[11]P = \mathcal{O}$.

Le résultat du calcul de $t(P, Q)$ est récapitulé dans le tableau (3.2). La dernière valeur de $f$ égale à $25 + 36i$. D'après la section (2.4.4), une manière simplifiée pour

| Etape $i$ | Op. | $V$ | $f$ | $f^2$ |
|---|---|---|---|---|
| init | – | $P = (31, 18)$ | 1 | 1 |
| $2 : m_2 = 0$ | DBL | $[2]P = (4, 38)$ | $23 + 18i$ | $33 + 11i$ |
| $1 : m_1 = 1$ | DBL | $[4]P = (13, 19)$ | $5 + 31i$ | $10 + 9i$ |
| $1 : m_1 = 1$ | ADD | $[5]P = (14, 36)$ | $18 + 32i$ | – |
| $0 : m_0 = 1$ | DBL | $[10]P = (31, 25)$ | $25 + 36i$ | – |
| $0 : m_0 = 1$ | ADD | $[11]P = \mathcal{O}$ | $25 + 36i$ | – |

TAB. 3.2 – Calcul de $t(P, Q)$

obtenir la valeur du couplage à partir de la valeur de $f$ est de mettre $t(P, Q) = s^{p-1}$ tel que $s = f^{(p+1)/m} = (25 + 36i)^4 = 38 + 11i$. Par conséquent, $t(P, Q) = s^{p-1} = (38 - 11)/(38 + 11i) = 7 + 9i$. Cette méthode est beaucoup plus efficace que celle qui consiste à mettre $t(P, Q) = f^{(p^2-1)/m} = (25 + 36i)^{160} = 7 + 9i$.

#### 3.4.5.2 Calcul de $t([3]P, Q)$

Posons $P' = [3]P = (23, 35)$. Le calcul de $t(P', Q)$ s'effectue de la même manière que celui de $t(P, Q)$. En voici les multiples du points $P'$ auxquels on a besoin durant le calcul du couplage selon le principe de la chaîne d'addition du nombre $m = 11$, $[2]P' = (14, 7)$, $[4]P' = (31, 18)$, $[5]P' = (13, 19)$, $[10]P' = (23, 8)$ et $[11]P' = \mathcal{O}$. La dernière valeur de $f = 24 + 10i$. Ainsi, $t(P', Q) = s^{p-1}$ tel que $s = f^4 = 3 + 42i$. En conséquence, $t(P', Q) = (3 - 42)/(3 + 42i) = 18 + 35i$.

Nous avons $t(P, Q)^3 = (7 + 9i)^3 = 18 + 35i = t([3]P, Q)$. Ceci est vérifié à cause de la propriété de la bilinéarité de l'application (le couplage de Tate) $t$. Le calcul de $t([3]P, Q)$ est récapitulé dans le tableau (3.3).

| Etape $i$ | Op. | $V$ | $f$ | $f^2$ |
|---|---|---|---|---|
| init | – | $P' = (23, 35)$ | 1 | 1 |
| $2 : m_2 = 0$ | DBL | $[2]P' = (14, 7)$ | $2 + 18i$ | $24 + 29i$ |
| $1 : m_1 = 1$ | DBL | $[4]P' = (31, 18)$ | $21 + 40i$ | $2 + 3i$ |
| $1 : m_1 = 1$ | ADD | $[5]P' = (13, 19)$ | $16 + 21i$ | – |
| $0 : m_0 = 1$ | DBL | $[10]P' = (23, 8)$ | $24 + 10i$ | – |
| $0 : m_0 = 1$ | ADD | $[11]P' = \mathcal{O}$ | $24 + 10i$ | – |

TAB. 3.3 – Calcul de $t([3]P, Q)$

## 3.5 Conclusion

Ce chapitre a montré le calcul du couplage de Weil ainsi que celui de Tate en faisant appel à l'algorithme de Miller. Celui-ci évalue, itérativement, la fonction d'un diviseur principal de degré zéro sur un point précis de la même courbe elliptique. Nous avons décidé de favoriser l'utilisation du couplage de Tate à celui de Weil pour sa rapidité de calcul et de proposer des améliorations au calcul du couplage de Tate afin de rendre celui-ci plus efficace. Le chapitre est terminé par un exemple concret sur les étapes de calcul du couplage de Tate et la vérification de sa propriété de bilinéarité.

# Chapitre 4

# Le chiffrement basé sur l'identité

En 2001, la cryptographie a réalisé un pas important lorsque Boneh et Franklin [BF01] dévoilaient concrètement le premier système de chiffrement basé sur l'identité en utilisant les couplages bilinéaires. Dans le souci d'élucider ce système, nous traitons dans ce chapitre les points suivants. La première section est réservée au problème du logarithme discret et les autres problèmes qui s'y réfèrent, tel que le Diffie-Hellman Bilinéaire. Dans la deuxième section, nous définissons les systèmes de chiffrement basés sur l'identité (IBE)[1] et leurs notions de sécurité. À travers la troisième section, nous dressons une comparaison d'un système IBE relativement à un système de chiffrement à clé publique usuel. Nous finalisons le chapitre par une présentation succincte du système IBE de Boneh et Franklin en se basant sur sa sécurité sémantique.

[1]Identity-Based Encryption

## 4.1 Le Problème du Logarithme Discret et les problèmes connexes

La sécurité de plusieurs cryptosystèmes repose sur la difficulté du problème de logarithme discret (DL)[2]. Nous définissons le problème (DL) et les problèmes qui lui sont liés, discutons les attaques les plus connues et décrivons certains protocoles standards qui sont basés sur ces problèmes.

### 4.1.1 Définitions

Soit $(\mathbb{G}, *)$ un groupe multiplicatif fini d'ordre $n$. On considère que $n$ est premier, ainsi, le groupe est cyclique et généré par un générateur $g \in \mathbb{G}$.

**Définition 4.1.1.** Soit $h \in \mathbb{G}$, tel que $h = g^x$ pour un certain $x \in \mathbb{Z}_n^*$ inconnu. Étant donnés $g$ et $h$, le problème du logarithme discret (DL) est de trouver $x$. Nous utilisons la notation $DL_g(h) = x$.

Un autre problème étroitement lié au problème de logarithme discret est le problème de Diffie-Hellman calculatoire.

**Définition 4.1.2.** Soient $a, b \in \mathbb{Z}_n^*$. Etant donnés $g, g^a, g^b$, le problème de Diffie-Hellman calculatoire (CDH) est de trouver l'élément $h \in \mathbb{G}$ tel que $h = g^{ab}$. Nous utilisons la notation $CDH_g(g^a, g^b) = h$.

Évidemment, le $CDH$ n'est pas plus difficile que le $DL$ ; la possibilité de calculer le logarithme discret conduit à la résolution du problème de Diffie-Hellman calculatoire. Cependant, l'inverse n'est généralement pas sûr. Maurer et Wolf [MW99] montrent que les deux problèmes sont équivalents si on peut construire une courbe elliptique sur $\mathbb{F}_n$ avec certaines propriétés.

En outre du problème de Diffie-Hellman calculatoire, il existe un problème plus facile connu sous le nom de Diffie-Hellman décisionnel.

[2]Discrete Logarithm

**Définition 4.1.3.** Soient $a, b, c \in \mathbb{Z}_n^*$. Étant donnés $g, g^a, g^b, g^c$, le problème de Diffie-Hellman décisionnel ($DDH$) est de décider si $g^c = g^{ab} (\mathrm{mod} n)$. On utilise la notation $DDH_g(g^a, g^b, g^c) = 1$ si $CDH(g^a, g^b) = g^c$ et $DDH_g(g^a, g^b, g^c) = 0$ sinon.

Il est clair que le problème de Diffie-Hellman calculatoire est au moins aussi difficile que sa version décisionnelle ; une méthode qui permet de calculer $h = CDH_g(g^a, g^b)$ et vérifie si $g^c = h$ résout ainsi le problème $DDH$. Réciproquement, pour la plupart des groupes, il n'est pas clair que le $DDH$ est plus facile que le $CDH$. Les groupes qui remplissent une telle propriété (c.-à-.d., le $CDH$ est difficile mais le $DDH$ est facile) sont appelés les groupes de Gap Diffie-Hellman (GDH). Les relations entre $DL$, $CDH$, et $DDH$ sont montrées comme suit.

$$DL \longrightarrow CDH \longrightarrow DDH$$

où $A \longrightarrow B$ signifie que le problème $A$ est au moins aussi difficile que le problème $B$.

### 4.1.2 Les attaques sur le problème du logarithme discret

Plusieurs attaques sont connues sur le problème du logarithme discret (voir [Odl00] et [Dou03]). Parmi celles-ci, nous citons les deux méthodes génériques *Baby-Step Giant-Step* de Shanks et *Rho* de Pollard. Le fait que ces méthodes n'exigent aucune propriété particulière des groupes, elles peuvent ainsi être appliquées à n'importe quel groupe. Toute méthode générique prend environ $O(\sqrt{n})$ opérations pour résoudre le problème du logarithme discret, où $n$ est l'ordre premier du groupe. Notons que la quantité de travail nécessaire pour résoudre le logarithme discret avec une méthode générique est exponentielle dans la taille des entrées. Cela fait que les groupes qui n'ont pas d'attaques connues, autres que les méthodes génériques, sont appropriés pour la construction de protocoles cryptographiques basés sur le logarithme discret.

### 4.1.3 Certains protocoles basés sur le $CDH$

Nous présentons ci-après, le protocole d'échange de clés de Diffie-Hellman et le chiffrement El-Gamal.

#### 4.1.3.1 Le Protocole d'Échange de clés de Diffie-Hellman

Afin de partager une clé secrète commune sur un canal non sécuritaire, deux participants $A$ et $B$, ne disposant d'aucun secret au préalable, peuvent utiliser le protocole d'échange de clés de Diffie-Hellman. Soit $\mathbb{G}$ un groupe d'ordre $n$ premier et $g$ son générateur ($G =< g >$). Le secret commun partagé par $A$ et $B$ serait un élément de $\mathbb{G}$. Le protocole est le suivant :

1. $A$ sélectionne aléatoirement un entier $a \in \mathbb{Z}_n^*$, calcule $g^a$ et transmet $g^a$ à $B$.
2. $B$ sélectionne aléatoirement un entier $b \in \mathbb{Z}_n^*$, calcule $g^b$ et transmet $g^b$ à $A$.
3. $A$ calcule $(g^b)^a$ et $B$ calcule $(g^a)^b$ ; la clé secrète partagée est $h = g^{ab}$.

La sécurité du protocole d'échange de clés de Diffie-Hellman est basée sur la difficulté du problème de Diffie-Hellman calculatoire. Précisément, un adversaire qui intercepte les messages $g^a$ et $g^b$ serait obligé de résoudre $CDH_g(g^a, g^b)$ pour trouver le secret $g^{ab}$, où $g$ est connu publiquement. Notons que ce protocole est exposé à l'attaque *man-in-the-middle*, car les participants $A$ et $B$ ne s'authentifient pas.

#### 4.1.3.2 Le chiffrement El-Gamal

Soit $\mathbb{G}$ un groupe généré par $g$. Dans le schéma de chiffrement El-Gamal, l'usager $A$ possède une clé privée $s \in \mathbb{Z}_n^*$ et une clé publique correspondante $h = g^s$. Ainsi, $B$

peut chiffrer un message $M$, qui est représenté comme étant un élément de $\mathbb{G}$, avec la clé publique $h$ de $A$ comme suit :

1. $B$ sélectionne aléatoirement un entier $r \in \mathbb{Z}_n^*$ et calcule $a = g^r$ et $b = h^r M$.
2. $B$ envoie le texte chiffré $(a, b)$ à $A$.
3. $A$ calcule $b/a^s = (h^r M)/g^{rs} = (h^r M)/h^r = M$

Le message $M$ est caché par le masque $h^r$, qui est uniformément distribué dans $\mathbb{G}^*$ car $r$ est choisi aléatoirement. En effet, le destinataire $A$ peut calculer le masque $a^s = g^{rs} = h^r$ puisqu'il connaît la clé secrète $s$. Un adversaire n'ayant pas cette connaissance a besoin de résoudre $CDH_g(a, h) = g^{rs} = h^r$ pour trouver le masque qui lui permet de déchiffrer le message. Ainsi, on dit que le schéma de chiffrement ElGamal est basée sur la difficulté du problème de Diffie-Hellman calculatoire.

### 4.1.4 Le logarithme discret sur les courbes elliptiques

Bien que la majorité des protocoles basés sur le logarithme discret avaient initialement été définis sur le groupe multiplicatif d'un corps fini $\mathbb{F}_q^*$, le problème du logarithme discret (et les autres problèmes qui lui sont liés) peut être utilisé sur n'importe quel autre groupe (multiplicatif). Aussi bien, ces protocoles peuvent être implantés sur des groupes qui offrent beaucoup plus de sécurité ou plus d'efficacité tels que les groupes de points des courbes elliptiques où on pense que le problème du logarithme discret est plus difficile que dans le corps $\mathbb{F}_q^*$. Par conséquent, les cryptosystèmes basés sur les courbes elliptiques peuvent offrir le même niveau de sécurité que ceux définis sur $\mathbb{F}_q^*$ en utilisant des clés plus courtes. Cependant, les courbes supersingulières qui sont adaptées pour l'implantation des cryptosystèmes (voir la section 2.4.4 du chapitre précédent) à cause de leur structure particulière, présentent

un inconvénient dans le sens où il existe une réduction efficacement calculable à des groupes où le logarithme discret peut être calculé en temps sous-exponentiel.

#### 4.1.4.1 La réduction de MOV

La réduction de MOV (Menezes, Okamoto et Vanstone) [MOV93] vise à réduire le problème du logarithme discret d'un groupe de points d'une courbe elliptique supersingulière, définie sur le corps $\mathbb{F}_q$, à un problème de logarithme discret dans le groupe multiplicatif de l'extension de corps $\mathbb{F}_{q^k}$ de degré $k$.

Soit $E/\mathbb{F}_q$ une courbe elliptique et $P$, $Q \in E(\mathbb{F}_q)$ des points de $m$-torsion sur la courbe tels que $Q = lP$. L'algorithme de MOV qui est utilisé pour réduire le logarithme discret de $P$ et $Q$ fait usage du couplage de Weil $e_m : E[m] \times E[m] \longrightarrow \mathbb{F}_{q^k}$ et est donné comme suit [KKSU00] [SU01] :

1. Déterminer le plus petit entier $k$ tel que $E[m] \subseteq E(\mathbb{F}_{q^k})$.
2. Trouver $s \in E[m]$ tel que $\alpha = e_m(P, s)$ a un ordre $m$.
3. Calculer $\beta = e_m(Q, s)$.
4. Calculer $l = DL_\alpha(\beta)$, le logarithme discret de $\beta$ en base $\alpha$ dans $\mathbb{F}^*_{q^k}$

Il en résulte de cette réduction que pour une meilleure sécurité, aussi bien dans le groupe de points de la courbe elliptique que dans le groupe de réduction correspondant, on doit choisir la taille des paramètres suffisamment grande dans les courbes elliptiques. Cette même structure est utilisée pour construire des groupes appelés Gap Diffie-Hellman.

### 4.1.5 Les groupes de Gap Diffie-Hellman

Dans cette section nous définissons les groupes de Gap Diffie-Hellman (GDH), expliquons comment ils peuvent être construits en utilisant les couplages bilinéaires et décrivons le problème de Diffie-Hellman bilinéaire.

#### 4.1.5.1 Les groupes de Gap Diffie-Hellman

Le problème de gap Diffie-Hellman résulte des deux problèmes de DH décisionnel et calculatoire. Soit $(\mathbb{G}, *)$ un groupe cyclique d'ordre $n$ premier et généré par $g$. Le problème de GDH est défini comme suit.

**Définition 4.1.4.** Soient $a, b \in \mathbb{Z}_n^*$. Étant donnés $g, g^a, g^b$, le problème de gap Diffie-Hellman ($GDH$) est de résoudre le problème de Diffie-Hellman calculatoire $CDH_g(g^a, g^b)$, probablement, à l'aide de l'oracle de Diffie-Hellman décisionnel.

Il est évident que le $GDH$ n'est pas plus difficile que le $CDH$. Naturellement, le gap Diffie-Hellman survient dans les groupes où le $CDH$ est difficile mais le $DDH$ est facile. Les groupes qui satisfont cette propriété sont appelés les groupes de Gap Diffie-Hellman.

#### 4.1.5.2 Réalisation avec les couplages

Soit un couplage bilinéaire non dégénéré $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$, où $\mathbb{G}_1$ est un groupe cyclique d'ordre $n$ premier, $\mathbb{G}_2$ est un groupe cyclique d'ordre $n$, où les opération peuvent être écrits multiplicativement. Montrons qu'un tel couplage permet de résoudre le problème de $DDH$ dans $\mathbb{G}_1$, donné par $DDH_P(aP, bP, cP)$ (décider si $abP = cP$). Précisément, en utilisant la bilinéarité du couplage $e$, on a :

$$e(aP, bP) = e(P, bP)^a = e(P, abP) = e(P, cP) \text{ si et seulement si } ab = c.$$

Ainsi, si un tel couplage existe, le problème de $DDH$ dans $\mathbb{G}_1$ est facile et le groupe $\mathbb{G}_1$ est un groupe $GDH$. Joux et Nguyen [JN01] montrent comment construire des courbes elliptiques où le $CDH$ est équivalent au $DL$ mais le $DDH$ est facile, permettant ainsi de représenter des groupes $GDH$.

### 4.1.6 Le problème de Diffie-Hellman Bilinéaire

La bilinéarité des couplages offre la possibilité de construire des applications cryptographiques. Précisément, la propriété de bilinéarité $e(aP, Q) = e(P, Q)^a = e(P, aQ)$ peut être utilisée pour véhiculer la valeur de $a$ (éventuellement un secret) à partir d'une coordonnée vers l'autre sans révéler la valeur réelle de $a$. Ainsi, la sécurité de la plupart des protocoles basés sur les couplages dépend d'un nouveau type de problème connu par le *problème de Diffie-Hellman Bilinéaire* (BDH)[3] (voir [Yac02] et [CL02] ). Soit $(\mathbb{G}_1, +)$ un groupe additif cyclique d'ordre $n$ et généré par $P$, $(G_2, *)$ un groupe multiplicatif lui aussi d'ordre $n$ et $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$ un couplage bilinéaire. Ainsi, $h = e(P, P)$ est un générateur de $\mathbb{G}_2$ et le problème de Diffie-Hellman bilinéaire est défini comme suit.

**Définition 4.1.5.** Soient $a, b, c \in \mathbb{Z}_n^*$. Étant donnés $P, aP, bP, cP$, le problème de Diffie-Hellman bilinéaire (BDH) est de calculer $e(P, P)^{abc}$. On utilise la notation $BDH_P(aP, bP, cP) = e(P, P)^{abc}$.

Certainement, le $BDH$ n'est pas plus difficile que le $CDH$. Précisément, si on peut obtenir $Q = CDH_P(aP, bP) = abP$, on pourrait facilement calculer $e(Q, cP) = e(P, P)^{abc}$. En outre, le $BDH$ dépend de la difficulté du $CDH$ dans $\mathbb{G}_2$. Ainsi, la difficulté du problème $BDH$ dépend de la difficulté du $CDH$ dans les deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$. Les relations entre les différents problèmes de Diffie-Hellman sont décrites

[3]Bilinear Diffie-Hellman

dans la figure ( 4.1).

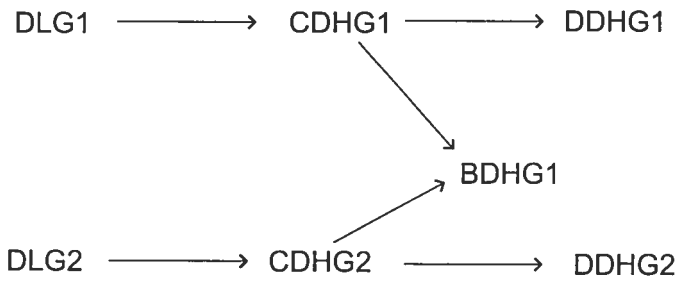



FIG. 4.1 – Les groupes GDH et les couplages ([CL02]

## 4.2 La cryptographie basée sur l'identité

Évidemment l'application la plus marquante des couplages est la réalisation de la cryptographie à clé publique basée sur l'identité (ID-PKC), le concept qui a été proposé par Shamir [Sch84] en 1984. Le but de Shamir était de soulager l'embarras des certificats qui sont essentiels dans les systèmes de chiffrement à clé publique conventionnels pour garantir l'authenticité des clés publiques.

Bien que l'idée de ID-PKC remonte à 1984, sa réalisation complète demeurait un problème ouvert durant plusieurs années. Dans son article original [Sch84], Shamir donne un exemple concret d'un schéma de signature basé sur l'identité. Le chiffrement basé sur l'identité, par contre, s'est avéré une tâche beaucoup plus difficile. En 2001, Boneh et Franklin [BF01] proposaient la première implémentation complète d'un système de chiffrement basé sur l'identité, en utilisant la bilinéarité des couplages. Nous présentons ce système dans la section 4.4.

### 4.2.1 Définitions

Dans cette partie, nous discutons de la cryptographie à clé publique en général, donnons la définition de la cryptographie basée sur l'identité, et définissons certaines notions en lien avec la cryptographie basée sur l'identité.

#### 4.2.1.1 La cryptographie à clé publique

Dans la cryptographie à clé publique, chaque utilisateur $U$ possède une paire de clés $(P_U, S_U)$ qui consiste en une clé publique et une clé secrète. la paire de clés est générée soit par l'utilisateur lui-même, ou par une certaine autorité centrale. Dans ce dernier cas, un canal sécuritaire est nécessaire pour acheminer cette paire de clés à l'utilisateur. La génération d'une paire de clés est effectuée, tout d'abord, en choisissant aléatoirement une clé secrète $S_U$. Puis on applique une fonction à sens unique à $S_U$ pour obtenir $P_U$.
Ainsi, les autres usagers ne peuvent pas obtenir la clé secrète à partir de la clé publique $P_U$. Par exemple, dans le système de chiffrement ElGamal, la clé secrète $s$ est choisie aléatoirement et la clé publique correspondante est définie par $h = g^s$, où $g$ est un générateur du groupe publiquement connu. Calculer $s$ à partir de $h$ et $g$ revient à calculer le logarithme discret.

Les autres participants utilisent la clé publique de l'utilisateur $U$ afin de lui envoyer des messages chiffrés, ou bien pour vérifier ses signatures. Ceci peut être effectué par n'importe qui, étant donné que la clé $P_U$ est une connaissance publique. D'autre part, pour déchiffrer un texte chiffré ou signer un message, $U$ utilise sa clé secrète $S_U$. Le secret du texte chiffré et l'authenticité de la signature proviennent alors du fait que l'utilisateur $U$ est le seul qui connaît $S_U$.

Cependant, le souci principal dans un schéma à clé publique est l'authenticité de la clé publique. Clairement, si une personne malveillante peut convaincre d'autres participants que la clé publique de $U$ est une certaine clé de son choix au lieu de la vraie clé $P_U$, il peut déchiffrer les messages envoyés à $U$ et forger les signatures sous le nom de $U$. Cependant, il est important que les participants dans un système PKC parviennent à vérifier l'authenticité des clés publiques des autres usagers.

La solution traditionnelle à ce problème d'authentification consiste à utiliser une infrastructure à clé publique (PKI). Une PKI fonctionne souvent avec un tiers de confiance appelé une *Autorité de Certification* (AC), qui peut garantir l'exactitude des clés publiques de la façon suivante : l'utilisateur $U$ s'identifie à l'autorité de certification (en suivant une certaine procédure d'authentification) et présente sa clé publique $P_U$. Ensuite, $U$ fournit une *preuve de possession* de la clé secrète correspondante ; ceci est souvent effectué par le biais d'une signature de la demande de certification avec la clé secrète. Si l'AC est convaincue que l'utilisateur $U$ possède réellement la clé secrète correspondante à la clé publique $P_U$, elle publie un certificat contenant l'identité de $U$, sa clé publique et possiblement d'autres informations, et signe toute cette information avec sa clé secrète. Les participants qui veulent communiquer secrètement avec $U$ doivent alors rechercher le certificat délivré par l'AC. Une signature valide émise par l'AC convaincra les autres usagers de l'authenticité de la clé publique $P_U$.

#### 4.2.1.2 La cryptographie basée sur l'identité

En 1984 Shamir [Sch84] a inventé le concept de la cryptographie basé sur l'identité qui s'adresse au problème de l'authenticité de différentes manières. Son idée originale avait pour objectif d'éviter le besoin pour l'authentification, en s'assurant que la valeur

réelle de la clé publique d'un utilisateur est dérivée directement à partir d'une information disponible publiquement qui identifie uniquement et incontestablement cet utilisateur. Cette information est désignée par une *identité* (digitale) d'un utilisateur. Dépendamment de l'application, l'identité peut être composée de (une ou une combinaison de) nom de l'utilisateur, numéro de l'assurance sociale, numéro de téléphone, adresse de courriel et possiblement d'autres informations personnelles. Dans ce cas, une clé publique d'un utilisateur est aisément disponible à n'importe qui connaissant son identité. Ainsi, on n'a pas besoin de rechercher sa clé publique. D'ailleurs, le fait qu'il n'y a pas de doute sur l'authenticité de la clé publique évite le recours à des certificats comme dans le cas des PKI. Cependant, la réalisation du lien entre les usagers et leurs identités digitales est loin d'être évident. Citons le cas où il y a plusieurs usagers qui possèdent le même nom, ... etc. Rappelons que dans les systèmes à clé publique conventionnels, la paire de clés est générée en choisissant aléatoirement une clé secrète et en appliquant une certaine fonction à sens unique pour calculer la clé publique. Dans le cas de la cryptographie basée sur l'identité, la paire de clés est obtenue différemment. Tout d'abord, La clé publique est déterminée uniquement à partir de l'identité de l'usager, au lieu d'être calculée à partir d'une clé secrète. Ainsi, la génération de clés ne peut pas être effectuée par les usagers eux-mêmes. Précisément, si un usager sait comment générer la clé privée correspondante à sa clé publique (son identité), il pourrait aussi générer les clés privées des autres usagers. Cependant, nous avons besoin d'un tiers appelé *Générateur de Clés Privées* (PKG)[4]. Après une certaine procédure d'authentification (similaire à l'authentification à l'AC dans le cas de PKI), le PKG génère la clé privée de l'usager. Le PKG détient ce privilège (la

[4]Private Key Generator

génération des clés privées) car il possède une information secrète communément appelée *la clé maître*. L'information correspondant à la clé maître et qui est disponible publiquement est appelée *les paramètres du système*. Une clé privée d'un usager est donc calculée par une fonction à sens unique de la clé publique et de la clé maître. Notons que ce schéma exige toujours un canal sécuritaire pour acheminer la clé privée à partir du PKG vers le bon usager.

Dans [BF01], Boneh et al. révèlent que la bilinéarité des couplages peut être utilisée pour transformer le système de chiffrement ElGamal en une variante du chiffrement basé sur l'identité.

### 4.2.2 La Sécurité des Systèmes de chiffrement basés sur l'Identité

Dans un schéma de chiffrement à clé publique, il y a plusieurs concepts de sécurité. Les plus importants sont la sécurité *sémantique* et la sécurité de *texte chiffré choisi*. Cette derniere est une notion de sécurité suffisamment forte et acceptable pour la plupart des applications y compris les systèmes de chiffrement traditionnels (non basés sur l'identité). Les notions de la sécurité sémantique et la sécurité du texte chiffré choisi sont décrites ci-après.

#### 4.2.2.1 La Sécurité sémantique

Un système de chiffrement à clé publique traditionnel est sémantiquement sécuritaire si un adversaire qui est donné un texte chiffré avec une clé publique connue n'apprend rien au sujet du texte clair correspondant.

Plus particulièrement, considérons le jeu suivant entre un challengeur et un adversaire :

1. l'adversaire reçoit une clé publique aléatoirement choisie par le challengeur
2. l'adversaire produit deux messages clairs de même taille $M_0$ et $M_1$ et reçoit, à partir du challengeur, le texte chiffré du message $M_b$, où $b$ est choisi aléatoirement dans $\{0, 1\}$,
3. l'adversaire produit (devine) $b'$ et gagne le jeu si $b = b'$.

Le système à clé publique est dit sémantiquement sécuritaire (IND-CPA)[5] s'il n'existe pas d'adversaire capable de gagner ce jeu en temps polynomial avec une probabilité non-négligeable.

#### 4.2.2.2 La Sécurité du texte chiffré choisi

Pour la définition de la sécurité du texte chiffré choisi, un jeu similaire est considéré, sauf que cette fois-ci, on attribue à l'adversaire une possibilité supplémentaire de pouvoir lancer au challenger un certain nombre de requêtes de déchiffrement. C'est-à-dire, l'adversaire présente au challenger un texte chiffré de son choix et le challenger lui renvoie le texte clair correspondant en utilisant la clé privée correspondante à la clé publique donnée à l'adversaire. La seule restriction est que l'adversaire ne peut pas lancer une requête de chiffrement sur le texte chiffré (produit par le challenger) sur lequel on veut le tester. Un système de chiffrement à clé publique standard (non basé sur l'identité) est appelé sécuritaire contre une attaque à texte chiffré choisi (IND-CCA)[6] s'il n'existe pas d'adversaire capable de gagner le jeu précédent avec une probabilité non-négligeable et en temps polynomial. La sécurité du texte chiffré choisi implique qu'étant donné un texte chiffré et une clé publique, un adversaire n'apprend rien au sujet du texte clair même s'il peut obtenir les textes clairs d'un certains nombre de

[5]IND-CPA : Indistinguishability against Chosen Plaintext Attack
[6]INDistinguishability against Chosen Ciphertext Attack

textes chiffrés de son choix.

Ces notions de sécurité doivent être renforcées pour satisfaire les besoins des systèmes de chiffrement basés sur l'identité, où toute valeur (comme une chaîne de caractères) peut servir comme étant une clé publique. En particulier, la clé publique n'est pas aléatoire. Ainsi, l'adversaire pourrait déjà avoir accès à un certain nombre de clés privées correspondant aux clés publiques de son choix. En outre, un système basé sur l'identité devrait être sécuritaire contre une attaque sur une clé publique (une identité) choisie par l'adversaire au lieu d'être aléatoire. Ci-après, nous présentons les notions de la sécurité sémantique et la sécurité du texte chiffré choisi pour les systèmes de chiffrement basés sur l'identité. Ces notions sont utilisées par Boneh et Franklin ( [BF01]) afin de prouver la sécurité de leur système IBE.

#### 4.2.2.3 La Sécurité sémantique d'un schéma basé sur l'identité

Considérons un adversaire et un challengeur dans un jeu similaire au IND-CPA précédent. Cependant, avec un schéma de chiffrement basé sur l'identité, l'adversaire peut choisir la clé publique (l'identité) sur laquelle il veut être testé, au lieu de recevoir une clé publique aléatoire à partir du challenger. En outre, l'adversaire est autorisé à effectuer un certain nombre de requêtes d'extraction de clés privées. C'est-à-dire, l'adversaire présente une clé publique (une identité) de son choix, au challenger, et obtient la clé privée correspondante. La seule contrainte est que l'adversaire ne peut pas lancer une requête d'extraction de clé privée sur laquelle on veut le tester. En particulier, cette identité ne devrait pas figurer dans la liste des requêtes d'extraction lancées précédemment. Un schéma basé sur l'identité serait sémantiquement sécuritaire (IND-ID-CPA) s'il n'existe pas d'adversaire pouvant gagner le jeu en temps polynomial et avec une probabilité non-négligeable plus que la moitié.

#### 4.2.2.4 La Sécurité du texte chiffré choisi d'un schéma IBE

Pour la sécurité du texte chiffré choisi dans un schéma de chiffrement basé sur l'identité, l'adversaire et le challenger jouent encore un jeu similaire à celui de IND-ID-CPA précédent. L'adversaire peut choisir l'identité sur laquelle il veut être testé. En plus, l'adversaire peut lancer des requêtes de déchiffrement comme dans le jeu IND-CCA. Dans ce cas, chaque requête de déchiffrement consiste en une paire de texte chiffré choisi et d'une identité. En réponse à cette requête, l'adversaire recevra, à partir du challenger, le texte clair correspondant. Évidemment, après le choix d'une identité et la réception d'un texte chiffré comme un challenge, un adversaire n'est plus autorisé à lancer une requête de déchiffrement sur cette combinaison en particulier. Un schéma basé sur l'identité est appelé sécuritaire contre une attaque de type *texte chiffré choisi* s'il n'existe pas d'adversaire capable de gagner le jeu en un temps polynomial et avec une probabilité non-négligeable supérieure à un demi (voir [BF01] pour une description formelle).

## 4.3 Comparaison avec la cryptographie à clé publique traditionnelle

La principale différence entre la cryptographie à clé publique basée sur l'identité (ID-PKC)[7] et la cryptographie à clé publique traditionnelle (PKI)[8] réside dans la manière de générer une paire de clés (privée, publique), et les moyens de les vérifier [PP03]. Dans une PKI traditionnelle, ceci est réalisé par le biais de certificats. Tandis que dans un système de chiffrement basé sur l'identité, le lien entre l'identité et la clé privée est géré par l'autorité de confiance PKG, alors que l'authenticité de

[7]Identity-based Public Key Cryptography
[8]Public Key Infrastructure

l'identité peut être vérifiée publiquement.

Dans ce qui suit, nous comparons les systèmes de chiffrement basés sur l'identité aux systèmes à PKI traditionnels, en examinant les aspects suivants : l'authenticité des paramètres (publiques) du système, l'enregistrement à une autorité, la révocation des clés, la distribution des clés, la sécurité de la clé maître, et enfin la délégation des clés privées.

### 4.3.1 L'authenticité des paramètres du système

Dans un système IBE, nous supposons qu'un attaquant génère sa propre clé maître et les paramètres système correspondants, et trompe les usagers en leur faisant croire que les paramètres du système "forgés" sont les vrais. Il s'en suit que, pour toute identité, il peut dériver la clé privée correspondant à sa clé maître. Cependant, il pourrait personnifier le PKG et générer des clés privées aux usagers en réponse à leurs requêtes. Dès lors, il peut déchiffrer tout message chiffré avec ces paramètres forgés. En conclusion, il est important que le PKG garantisse, d'une manière ou d'autre, l'authenticité des vrais paramètres du système.

Un problème similaire peut se produire dans une situation de PKI, où les usagers ont besoin d'être sûrs de l'authenticité de la clé publique de l'autorité de certification. Précisément, si un attaquant peut faire croire aux usagers qu'une certaine clé publique de son choix est la clé publique de l'AC[9], il pourrait alors créer des certificats contenant les clés publiques forgés pour lesquelles il possède la clé secrète. Par conséquent, l'attaquant pourrait lire les messages chiffrés par les clés publiques forgées et créer des signatures qui ont l'air d'être valides. La seule différence avec le schéma IBE est que l'attaquant ne peut pas personnifier l'AC, car les usagers se rendront compte que

[9]Autorité de Certification

leur certificat demandé contient une clé publique incorrecte.

### 4.3.2 Enregistrement à une autorité

Dans les deux systèmes, un usager qui veut participer a besoin de s'enregistrer à une autorité d'enregistrement[10], qui est souvent considérée comme étant une partie de l'AC, respectivement le PKG. Après l'opération d'authentification, l'autorité d'enregistrement choisit une identité numérique unique à l'usager, par exemple sous forme d'une adresse de courriel. Dans un système à PKI, un usager peut ainsi présenter son identité et une clé publique à l'AC avec une preuve de possession de la clé secrète correspondante. L'AC publie ensuite un certificat contenant l'identité numérique et la clé publique. Similairement, dans un schéma IBE, l'usager présente son identité au PKG. L'autorité d'enregistrement est responsable de l'unicité de l'identité numérique et le lien entre l'identité et l'usager. Le PKG génère ensuite la clé privée correspondante à cette identité.

Les systèmes basés sur l'identité exigent un canal sécuritaire pour transmettre la clé privée du PKG à l'usager. Cependant, les systèmes IBE semblent fonctionner mieux dans les applications où il est facile de réaliser un canal sécuritaire, ou là où les usagers demandent moins souvent les clés privées.

### 4.3.3 La révocation des clés

Dans un schéma à PKI, lorsqu'un certificat est révoqué, les autres usagers sont informés par le biais d'une liste de révocation de certificats (CRL)[11]. Par exemple,

[10] Registration Authority
[11] Certificate revocation list

cela se produit lorsque un usager quitte un groupe de travail ou qu'une clé privée est compromise. Dans ce dernier cas, lorsqu'une paire de clés a besoin d'être remplacée après l'expiration du certificat[12], un usager peut tout simplement générer une nouvelle paire de clés et obtenir un certificat. Ainsi, la liste de révocation peut bien être utilisée dans un schéma IBE. Cependant, étant donné que la clé publique d'un usager est dérivée à partir de son identité, il ne peut pas changer d'identité à chaque fois qu'une nouvelle clé privée est exigée. Une solution proposée dans [BF01] consiste à concaténer l'identité avec une certaine information publiquement connue. Par exemple, lorsque l'année courante est ajoutée à l'identité, les usagers peuvent seulement utiliser leurs clés durant l'année en cours. Ainsi, les clés privées expirent annuellement et chaque usager devrait demander une autre clé privée chaque année. Mais, que se passe-t-il si une clé privée vient d'être compromise ? Dans ce cas, l'usager devrait attendre jusqu'à la fin de l'année pour pouvoir avoir une autre clé privée. Cette situation pourrait être beaucoup plus rudimentaire en concaténant, par exemple, l'identité avec la date en cours au lieu de l'année en cours. Avec cette approche, la révocation de clés est plus simple ; lorsque un usager quitte la compagnie et sa clé privée a besoin d'être révoquée, la compagnie informe le PKG de ne plus générer de clés pour cet usager.

### 4.3.4 La distribution des clés

Incontestablement, le plus grand apport de la cryptographie basée sur l'identité est la simplification de la distribution des clés (publiques). À savoir, toutes les clés publiques peuvent être dérivées à partir de l'identité des usagers. Ainsi, l'obtention de la clé publique d'un autre usager, pour un chiffrement ou une vérification de signature, devient une procédure plus simple et plus transparente. En revanche, dans un schéma

[12]key rollover

à PKI, on doit rechercher la signature de l'AC et la date d'expiration qui pourrait être accomplie en utilisant une CRL mise à jour.

### 4.3.5 La sécurité de la clé maître

Dans un schéma IBE, le PKG représente le seul point de faiblesse. Un attaquant qui est capable de trouver la clé maître du PKG pourrait générer toutes les clés privées, et subséquemment, lire tous les messages chiffrés. Il est ainsi extrêmement important pour le PKG de garder secrètement la clé maître. Afin d'empêcher la sauvegarde de la clé maître dans un seul endroit, [BF01] propose d'utiliser le calcul multipartie, en se servant de la caractéristique de la bilinéarité des couplages, pour distribuer la clé maître sur plusieurs PKGs. Similairement, un attaquant qui peut obtenir la clé secrète d'une autorité de certification aurait la possibilité de créer des signatures au nom de l'AC pour les nouvelles clés publiques de son choix. Cependant, la connaissance de la clé secrète de l'AC n'a aucun effet sur les messages chiffrés sous les clés publiques précédentes.

### 4.3.6 Délégation des clés

L'une des applications de la cryptographie basée sur l'identité est la délégation de clés de déchiffrement (voir [BF01]). Supposons qu'un usager $U$ qui joue en même temps le rôle d'un PKG, donc il a la possibilité d'extraire les clés privées de son choix, veut partir en voyage pour quelques jours, il installe sur son *laptop* les clés privées de ces jours-ci et non pas la clé maître. Si le laptop est volé, seulement les clés privées qui y sont installées sont compromises et non pas la clé maître qui reste secrète.
Pour une autre application, l'usager $U$ peut déléguer d'autres usagers à utiliser la *ligne d'objet* d'un courriel comme étant l'identité servant au chiffrement. Supposons

que $U$ a plusieurs employés sous sa responsabilité, chacun est responsable d'une tâche différente (achats, comptabilité, ...). $U$ donne à chaque employé une clé privée correspondant à son identité qui est en lien avec sa tâche. De cette manière, $U$ est sûr que chaque employé peut seulement déchiffrer le courriel qui lui est destiné.

## 4.4 Réalisation avec les couplages

Dans cette section, nous présentons le schéma de chiffrement basé sur l'identité (IBE) de Boneh et Franklin [BF01].

### 4.4.1 Le schéma BasicIdent

le schéma *BasicIdent* de IBE proposé par Boneh et Franklin dans [BF01] consiste en quatre algorithmes : *Setup*, *Extract*, *Encrypt*, et *Decrypt*. L'algorithme *Setup* est exécuté par le PKG (le générateur de clés privées) pour générer la clé maître et les paramètres (publiques) du système. Ceci est effectué sur l'entrée d'un paramètre de sécurité $k$, qui spécifie la taille , en nombre de bits, des groupes. L'algorithme *Extract* est exécuté, lui aussi, par le PKG à l'entrée d'une identité d'un usager, pour générer la clé privée correspondant à cette identité. Tout comme avec la cryptographie à clé publique, l'algorithme *Encrypt* prend en entrée un texte clair et une clé publique (une identité) et produit un cryptogramme. Similairement, *Decrypt* permet au détenteur de la clé privée correspondante de déchiffrer le cryptogramme qui lui est destiné. La version BasicIdent de IBE est donnée comme suit :

**Setup :** Sur l'entrée d'un paramètre de sécurité $k$, l'algorithme procède comme suit :

1. générer un nombre premier aléatoire de $k$ bits, deux groupes $(\mathbb{G}_1, +)$,

$(\mathbb{G}_2, *)$ d'ordre $q$, et un couplage bilinéaire $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$. Choisir un générateur arbitraire $P \in \mathbb{G}_1$.

2. Sélectionner un nombre aléatoire $s \in \mathbb{Z}_q^*$ et mettre $P_{pub} = sP$.
3. Choisir des fonctions de hachage cryptographiques $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}_1^*$ et $H_2 : \mathbb{G}_2 \longrightarrow \{0,1\}^n$ pour un certain entier $n$.

L'espace des messages est $\mathcal{M} = \{0,1\}^n$ et l'espace des cryptogrammes est $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n$. Les paramètres du système sont $params =< q, \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_{pub}, H_1, H_2 >$. La clé maître est $s \in \mathbb{Z}_q^*$.

**Extract :** Pour une identité (une chaîne de caractères) $ID \in \{0,1\}^*$, l'algorithme fonctionne comme suit :

1. Calculer $Q_{ID} = H_1(ID) \in \mathbb{G}_1^*$.
2. Mettre la clé privée $d_{ID}$ comme étant $d_{ID} = sQ_{ID}$ où $s$ est la clé maître.

**Encrypt :** Pour chiffrer un message $M \in \mathcal{M}$ en utilisant la clé publique $ID$, l'algorithme fonctionne comme suit :

1. Calculer $Q_{ID} = H_1(ID) \in \mathbb{G}_1^*$.
2. Choisir un nombre aléatoire $r \in \mathbb{Z}_q^*$.
3. Mettre le cryptogramme comme étant $C =< rP, M \oplus H_2(g_{ID}^r) >$ où $g_{ID} = e(Q_{ID}, P_{pub}) \in \mathbb{G}_2^*$.

**Decrypt** Soit $C =< U, V > \in \mathcal{C}$ le cryptogramme chiffré en utilisant la clé publique $ID$. Pour déchiffrer $C$ en utilisant la clé privée $d_{ID} \in \mathbb{G}_1^*$, on effectue le calcul : $V \oplus H_2(e(d_{ID}, U)) = M$.

Notons que dans la phase de chiffrement, le message $M$ est en *ou exclusif* bit-à-bit avec le masque $H_2(g_{ID}^r)$ pour former la composante $V$. Dans la phase de déchiffrement,

cette même composante $V$ est en ou exclusif avec le masque $H_2(e(d_{ID}, U))$. En utilisant la propriété de bilinéarité du couplage $e$, on a

$$e(d_{ID}, U) = e(sQ_{ID}, rP) = e(Q_{ID}, P)^{sr} = e(Q_{ID}, P_{pub})^r = g_{ID}^r$$

Ainsi, les deux masques de chiffrement et de déchiffrement sont les mêmes. La bilinéarité est utilisée pour transporter le secret $s$ de la première coordonnée à la deuxième, sans connaître la valeur réelle de $s$. Cette propriété est essentielle dans ce schéma basé sur l'identité, vu qu'elle permet à la clé maître $s$ de demeurer secrète, alors que les clés privées $d_{ID} = sQ_{ID}$ sont suffisantes pour que les usagers puissent déchiffrer leurs messages.

La sécurité de BasicIdent est considérée dans le modèle de l'oracle aléatoire, c'est-à-dire, l'analyse de la sécurité considère les deux fonctions de hachage $H_1$ et $H_2$ comme étant des oracles aléatoires. La sécurité de BasicIdent est basée sur le problème de Diffie-Hellman Bilinéaire (BDH)[13]. Précisément, [BF01] démontre que si le BDH est difficile, alors BasicIdent est un schéma de chiffrement basé sur l'identité sémantiquement sécuritaire (IND-ID-CPA). Ceci est effectuée par une démonstration par l'absurde ; un adversaire qui est capable de gagner le jeu IND-ID-CPA avec une probabilité non-négligeable de plus d'un demi peut être utilisé pour construire un algorithme qui est capable de résoudre le problème de BDH avec une probabilité non-négligeable. D'où le théorème suivant :

**Théorème 4.4.1.** *(Boneh et Franklin [BF01], théorème 4.1) Supposons que les deux fonctions de hachage $H_1$ et $H_2$ sont des oracles aléatoires. Alors* ***BasicIdent*** *est un schéma de chiffrement basé sur l'identité sémantiquement sécuritaire (IND-ID-CPA) en supposant que le BDH est difficile dans les groupes générés par $\mathcal{G}$. Concrètement, supposons qu'il existe un adversaire IND-ID-CPA $\mathcal{A}$ qui a une probabilité $\epsilon(k)$ contre*

[13]Bilinear Diffie-Hellman

*le schéma* ***BasicIdent****. Supposons que $\mathcal{A}$ effectue au plus $q_E > 0$ requêtes d'extraction de clés privées et $q_{H_2} > 0$ requêtes de hachage à la fonction $H_2$. Alors il existe un algorithme $\mathcal{B}$ qui résout le BDH dans les groupes générés par $\mathcal{G}$ avec une probabilité au moins égale à :*

$$Prob_{\mathcal{G},\mathcal{B}}(k) \geq \frac{2\epsilon(k)}{e(1+q_E)\cdot q_{H_2}}$$

*Où $e \approx 2.71$ est la base du logarithme naturel. Le temps d'exécution de $\mathcal{B}$ est $\mathbf{O}(temps(\mathcal{A}))$.*

### 4.4.2 Le schéma FullIdent

Lorsqu'un adversaire connaît le déchiffrement de certains textes chiffrés connus, la sécurité de IND-ID-CPA n'est pas suffisante. On aura plutôt besoin d'une notion de sécurité plus forte telle que IND-ID-CCA. Cependant, Boneh et Franklin [BF01] utilisent une technique de Fujisaki et Okamoto [FO99] afin de convertir le schéma BasicIdent précédent en FullIdent ; un schéma de chiffrement basé sur l'identité IND-ID-CCA sécuritaire. Le résultat suivant montre comment convertir un schéma IND-ID-CPA sécuritaire en un schéma IND-ID-CCA sécuritaire.

**Théorème 4.4.2.** *(Fujisaki et Okamoto [FO99], Théorème 14) Soit $\varepsilon$ un schéma de chiffrement à clé publique IND-CPA sécuritaire et soit $\varepsilon_{pk}(M;r)$ le chiffrement de $M$ en utilisant la chaîne de bits aléatoires $r$ sous la clé publique $pk$. Alors le schéma hybride $\varepsilon^{hy}$ défini par*

$$\varepsilon^{hy}_{pk}(M) =< \varepsilon_{pk}(\sigma; H(\sigma, M)), G(\sigma) \oplus M >$$

*est IND-CCA sécuritaire. Où $\sigma$ est une chaîne aléatoire choisie à partir d'un domaine approprié et $G$ et $H$ sont des fonctions de hachage.*

Boneh et Franklin appliquent cette transformation au schéma BasicIdent afin d'obtenir le schéma FullIdent suivant.

**Setup :** Comme dans le schéma BasicIdent. En plus, on sélectionne deux fonctions de hachage $H_3 : \{0,1\}^n \times \{0,1\}^n \longrightarrow \mathbb{Z}_q^*$ et $H_4 : \{0,1\}^n \longrightarrow \{0,1\}^n$

L'espace des messages est $\mathcal{M} = \{0,1\}^n$ et l'espace des cryptogrammes est $\mathcal{C} = \mathbb{G}_1^* \times \{0,1\}^n \times \{0,1\}^n$. Les paramètres du système sont $params =< q, \mathbb{G}_1, \mathbb{G}_2, e, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$. La clé maître est $s \in \mathbb{Z}_q^*$.

**Extract :** Comme dans le schéma BasicIdent.

**Encrypt :** Pour chiffrer un message $M \in \mathcal{M}$ en utilisant la clé publique $ID$, l'algorithme fonctionne comme suit :

1. Calculer $Q_{ID} = H_1(ID) \in \mathbb{G}_1^*$.
2. Choisir une chaîne aléatoire $\sigma \in \{0,1\}^n$.
3. Mettre $r = H_3(\sigma, M)$. Le cryptogramme est donné par :

$$C =< rP, \sigma \oplus H_2(g_{ID}^r), M \oplus H_4(\sigma) > \text{ où } g_{ID} = e(Q_{ID}, P_{pub}) \in \mathbb{G}_2^*.$$

**Decrypt** Soit $C =< U, V, W > \in \mathcal{C}$ le cryptogramme chiffré en utilisant la clé publique $ID$. Pour déchiffrer $C$ en utilisant la clé privée $d_{ID} \in \mathbb{G}_1^*$, l'algorithme fonctionne comme suit :

1. Calculer $V \oplus H_2(e(d_{ID}, U)) = \sigma$.
2. Calculer $W \oplus H_4(\sigma) = M$.
3. Mettre $r = H_3(\sigma, M)$. Vérifier si $U = rP$. Sinon, refuser le texte chiffré.
4. Produire $M$ comme étant le déchiffrement de $C$.

Dans la phase de chiffrement, le message $M$ est *masqué* par la valeur aléatoire $\sigma$. En outre, $\sigma$ est en ou exclusif bit-à-bit avec le masque $H_2(g_{ID}^r)$ pour former $V$. Dans la phase de déchiffrement, $V$ est en ou exclusif avec $H_2(e(d_{ID}, U))$. Cela génère la valeur de $\sigma$, car la valeur du couplage $g_{ID}^r = e(d_{ID}, U)$ est la même que celle du schéma

BasicIdent. Enfin le masque $H_4(\sigma)$ peut être supprimé à partir de $W = M \oplus H_4(\sigma)$ pour reproduire le message $M$.

Nous remarquons que la première partie du cryptogramme $< rP, \sigma \oplus H_2(g_{ID}^r) >$ correspond au chiffrement BasicIdent de $\sigma$ en utilisant la chaîne de bits aléatoires $r = H_3(\sigma, M)$. En outre, la dernière partie $M \oplus H_4(\sigma)$ est un chiffrement *à masque jettable* du message $M$ avec les bits aléatoires $H_4(\sigma)$. D'un autre coté, remplaçons $H_3$ et $H_4$ par les deux fonctions de hachage $H$ et $G$ respectivement, nous constatons que FullIdent est un schéma hybride à l'instar du schéma annoncé dans le théorème (4.4.2).

La sécurité de FullIdent est considérée dans le modèle de l'oracle aléatoire, c'est-à-dire, les fonctions de hachage $H_1$, $H_2$,$H_3$, $H_4$ sont perçues comme des oracles aléatoires. Boneh et Franklin [BF01] montrent que, sous l'hypothèse que le $BDH$ est un problème difficile, FullIdent est un schéma basé sur l'identité sécuritaire contre une attaque de type texte chiffré choisi (IND-ID-CCA). Précisément, si un adversaire gagne le jeu IND-ID-CCA avec une probabilité non négligeable plus que la moitié, alors on peut construire un algorithme qui est capable de résoudre le problème du $BDH$ avec une probabilité non négligeable. Ceci est étalé dans le prochain théorème.

**Théorème 4.4.3.** *(Boneh et Franklin [BF01], théorème 4.4 et 4.5) Supposons que les fonctions de hachage $H_1, H_2, H_3, H_4$ sont des oracles aléatoires. Supposons qu'un adversaire $\mathcal{A}$ gagne le jeu IND-ID-CCA dans le schéma FullIdent avec une probabilité $1/2 + \epsilon$ pour un $\epsilon$ non négligeable et $\mathcal{A}$ s'exécute, au plus, en temps $t$. Supposons que $\mathcal{A}$ effectue au plus $q_E > 0$ requêtes d'extraction de clés privées et $q_{H_2}, q_{H_3}, q_{H_4}$ requêtes aux fonctions de hachage $H_2, H_3, H_4$ respectivement. Alors il existe un algorithme $\mathcal{B}$ capable de résoudre le $BDH$ avec une probabilité au moins*

$$\frac{1}{q_{H_2}(q_{H_3}+q_{H_4})}[(\frac{\epsilon}{1+q_E+q_D}+1)(1-2/q)^{q_D}-1]$$

*Le temps d'exécution de $\mathcal{B}$ est au plus $t + O((q_{H_4} + q_{H_3}) \cdot n)$. Avec $e \approx 2.71$ est la base du logarithme naturel. $n$ est la taille de $\sigma$ et $q$ est l'ordre des groupes $\mathbb{G}_1$ et $\mathbb{G}_2$.*

Après la présentation du système de chiffrement basé sur l'identité (IBE) proposé par Boneh et Franklin [BF01], nous abordons dans le prochain chapitre l'environnement Java 2 micro-edition (J2ME) et son utilisation pour l'implantation des applications de la messagerie texte mobile SMS. Cela représente également l'outil que nous utiliserons pour implanter le système de chiffrement FullIdent.

# Chapitre 5

# la messagerie SMS dans J2ME

Java a pris pieds dans de nombreux domaines, particulièrement celui de l'informatique embarquée. Sa version J2ME (Java 2 Micro Edition) consiste en la technologie, APIs, les outils et les normes requis pour créer des applications destinées aux appareils ayant des capacités limitées de stockage et traitement tels que les téléphones mobiles et les assistants personnels. Il fournit ainsi une solution complète pour créer des produits dynamiques et extensibles. Le présent chapitre a pour objectif d'expliquer l'architecture de J2ME et le développement des applications sous cette plate-forme pour les téléphones mobiles. Tout en ayant à l'esprit que l'objectif primordial de ce projet a trait à la messagerie texte mobile (SMS), nous allons surtout mettre l'accent sur les outils et APIs de J2ME qui traitent ce domaine. Pour de plus ample information sur ce sujet, on suggère [Yua04], [Tre02] et les manuels de JCP[1] [Pro03].

[1] Java Community Process

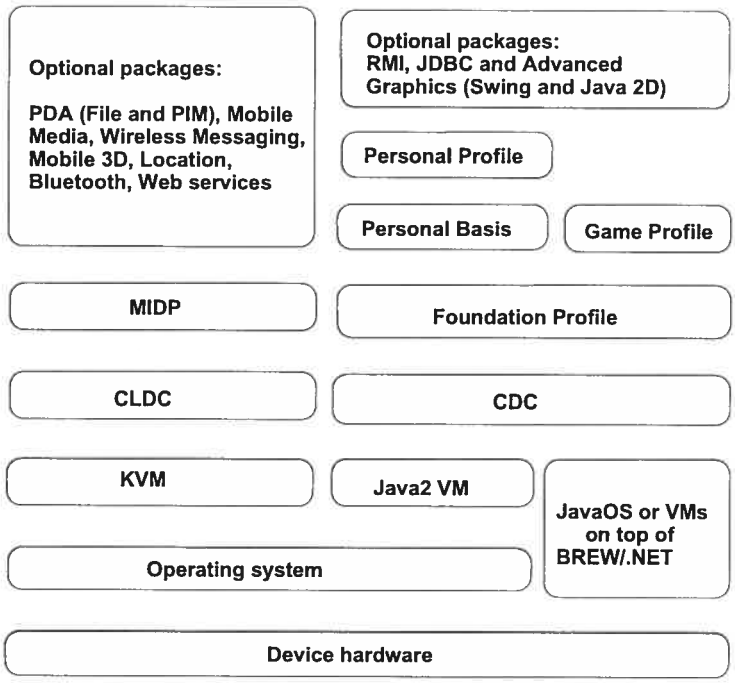



FIG. 5.1 – Les composants de J2ME [Yua04]

## 5.1 Modules de J2ME

J2ME fournit une gamme de technologies de machines virtuelles optimisées pour les dispositifs limités en termes de ressources. Pour ce genre de dispositifs, J2ME propose une configuration minimale de la machine virtuelle Java (JVM) et des APIs appropriés. Pour chaque type d'appareil, ces configurations peuvent être étendues avec de nouveaux APIs ou par des programmes utilisateurs J2ME (MIDlets). Ainsi, l'architecture de J2ME est composé de plusieurs couches, comme le montre la figure 5.1 :

**La machine virtuelle Java (JVM) :** Cette couche est une implémentation d'une JVM adaptée pour un type d'appareil particulier.

**La couche Configuration :** Définit la plate-forme minimale pour une catégorie *horizontale* d'appareils ayant des capacités de mémoires et traitements similaires. Elle définit, entre autres, les dispositifs de la JVM et les classes et librairies minimales requises par une même catégorie d'appareils. Par exemple, une configuration fournit les APIs génériques Java d'entrée/sortie, réseau et de la sécurité d'une MIDlet [Whi01].

**La couche Profiles :** Représente la couche au-dessus de la couche de *configuration*. Un profile définit les conditions spécifiques d'une certaine catégorie *verticale* de dispositifs. Le but principal d'un profile est de définir une plate-forme standard de Java pour une certaine famille de dispositifs et garantir l'interopérabilité entre eux. Typiquement, les profiles incluent les bibliothèques de classes spécifiques au sein d'une même famille de dispositifs. Par exemple, la gestion des événements et les fonctionalités de l'interface utilisateur.

### 5.1.1 Les Configurations CLDC

Rappelons que les configurations définissent les caractéristiques de base d'un environnement d'exécution pour un certain nombre de machines possédant un ensemble de caractéristiques et de ressources similaires. Ainsi, J2ME connaît actuellement deux configurations ; CLDC (Connected Limited Device Configuration) et CDC (Connected Device Configuration). La CLDC concerne les appareils possédant des ressources faibles comme les téléphones cellulaires (moins de 512ko de RAM, faible vitesse de processeur, connexion réseau limitée et intermittente [Pro00]) et une interface utilisateur réduite. Elle est utilisée sur une machine virtuelle appelée KVM (Kilo Virtual Machine). La CDC concerne les appareils possédant des ressources plus importantes (au moins 2 méga octets de RAM, un processeur 32 bits, ...) comme les assistants personnels. Elle s'utilise sur une machine virtuelle appelée CVM (Compact Virtual Machine). Comme notre système de chiffrement vise les appareils téléphoniques mobiles, nous nous limitons à étudier la configuration CLDC.

L'API de la CLDC se compose des packages suivants :

**java.io :** implémente les classes pour la gestion des entrées/sorties par flux.

**java.lang :** Les classes de base du langage Java (les types de données Integer, ...)

**java.util :** Les classes utilitaires, notamment pour gérer les collections, la date et l'heure.

**javax.microedition.io :** Les classes pour gérer des connections génériques.

La version actuelle de la CLDC ne permet pas la gestion des nombres flottants.

### 5.1.2 Les Profils MIDP

Les profils se composent d'un ensemble d'API particulier à une famille d'appareils ayant une certaine caractéristique commune. Ainsi, le profil MIDP (Mobile Information Device Profile), combiné avec la configuration CLDC, offre un environnement *runtime* de Java pour les appareils mobiles tels que les téléphones mobiles. Les spécifications du MIDP ont été définies à travers le programme de JCP[2]. L'objectif principal d'un MIDP est le développement d'applications sur des machines aux ressources et interfaces limitées. L'API du MIDP comporte les packages suivants :

**javax.microedition.midlet :** définit le cycle de vie d'une application MIDP (une midlet).

**javax.microedition.lcdui :** fournit les outils nécessaires pour le développement d'une interface utilisateur.

**javax.microedition.rms :** définit la persistance (le stockage permanent) de données.

## 5.2 Une application J2ME : une MIDlet

Les applications développées pour les appareils mobiles sont appelées des **MIDlets**. A l'instar des Applets, les MIDlets sont contrôlées par le programme qui les a démarré. Dans le cas d'une Applet, ce programme est le navigateur ou encore l'outil Appletviewer, et dans le cas d'une MIDlet, le programme en question est l'AMS (Application Management Software) qui est un environnement sur lequel une MIDlet est démarrée, arrêtée et désinstallée.

[2]Java Community Process

### 5.2.1 Cycle de vie d'une MIDlet

Une MIDlet passe par un cycle de vie bien défini qui consiste en trois phases : en pause, actif et détruit. Une fois créée, l'état d'une MIDlet est *en pause*, puis le système (l'AMS) invoque la méthode **startApp()** qui change l'état en actif. Le retour à l'état *en pause* se produit lors de l'invocation de la méthode **pauseApp()**. Finalement, à tout moment, l'exécution de la méthode **destroyApp()** conduit la MIDlet à l'état détruit. Le cycle de vie d'une MIDlet est montré sur la figure 5.2

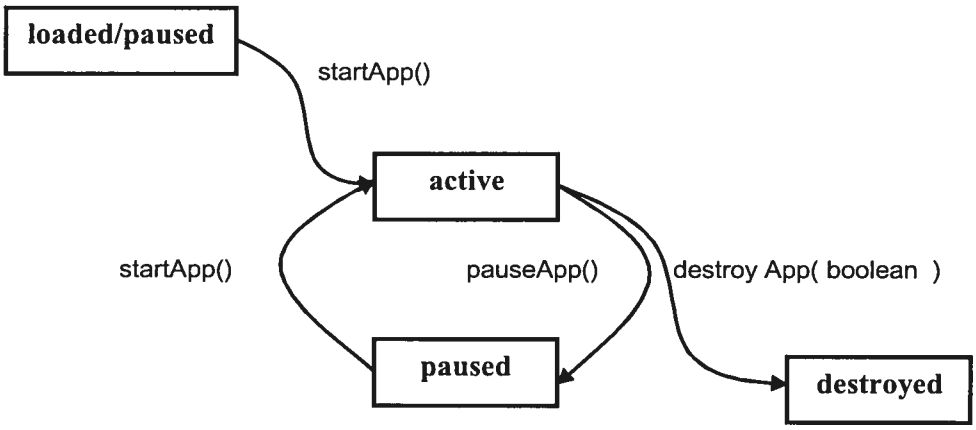



FIG. 5.2 – Cycle de vie d'une MIDlet

### 5.2.2 L'interface utilisateur

La réalisation d'interfaces graphiques pour les téléphones cellulaires diffère largement de celle des ordinateurs. Ceci est dû principalement à la différence des dimensions des deux écrans et leurs résolutions. Ainsi, dans J2ME, l'ensemble des outils graphiques se trouvent dans le package *javax.microedition.lcdui*. La classe Display est

considérée comme étant la plus importante. Elle permet en fait de modifier le contenu de l'interface. Pour obtenir une instance de cette classe, il est nécessaire d'employer la méthode statique getDisplay(MIDlet midlet). Il est à noter qu'un seul objet de type *Displayable* peut être activé à la fois.

### 5.2.3 Bouncy Castle et JAVA

Les librairies de Bouncy Castle[3] sont, entre autres, une implémentation en Java des algorithmes cryptographiques, qui ressemble quelque peu au JCE (Java Cryptography Extension) de Sun. Ces librairies sont destinées à être utilisés (importées) par des programmes Java (et J2ME). Comme J2ME est une plateforme minimale ne disposant pas de tels algorithmes cryptographiques de base, l'apport de Bouncy Castle est de très grande importance. Clairement, nous pouvons importer toute classe de Bouncy Castle jugée utile pour une application (eg. la classe BigInteger). Ceci est souvent accompagné de l'opération d'obfuscation qui élimine toutes les classes non utilisées avant de générer le byte code. Bouncy Castle implémente, entre autres, les fonctions de hachages usuelles (eg. MD5, SHA1) les chiffrement symétriques (eg. DES) et quelques chiffrements a clé publique. Cependant, le chiffrement basé sur l'identité n'y existe pas. On y trouve également une implémentation de base des courbes elliptiques. Concernant notre projet, nous avons exploité l'implémentation des courbes elliptiques, la classe BigInteger pour générer de grands nombres premiers et la fonction de hachage standard SHA1 qui sont disponibles dans les packages de Bouncy Castle.

[3]accessible au www.bouncycastle.org

## 5.3 La messagerie SMS

SMS (Short Message Service) est un système qui permet d'envoyer et recevoir sur un téléphone mobile GSM un message court d'environ 160 caractères. [4] Actuellement, sous certaines conditions, il est également possible d'envoyer des messages SMS à partir d'un ordinateur.

Le réseau GSM est capable de sauvegarder les message SMS pour un acheminement différé. Cela veut dire que si le téléphone mobile destinataire est éteint ou hors de la zone de couverture, les messages qui lui sont destinés sont sauvegardés de telle sorte qu'on peut les recevoir lors de la mise en marche de l'appareil.

Dans cette section, nous montrons les principales APIs de la messagerie sans-fil (WMA)[5], plus particulièrement celles qui sont en lien avec l'envoi et la réception des messages SMS.

### 5.3.1 Les APIs de la messagerie sans-fil

Les WMA de J2ME spécifient un ensemble d'APIs standards permettant aux applications J2ME s'exécutant sur des appareils téléphoniques sans-fil de communiquer aux abonnés du réseau via le protocole de SMS.

La propriété importante de WMA est cependant le fait qu'il permet aux appareils J2ME d'exécuter des applications serveurs basés sur le SMS. On pourrait utiliser un serveur SMS pour traiter et répondre automatiquement aux messages SMS reçus par une application J2ME. Enfin, contrairement aux serveurs HTTP qui sont basés sur des adresses IP, les adresses des serveurs SMS sont identifiées par leurs numéros de

[4] lorsque un message SMS dépasse 160 caractères, il est possible de le diviser en plusieurs segments avant de l'envoyer

[5] Wireless Messaging API

téléphone respectifs.

### 5.3.2 Les classes de WMA

On peut accéder aux services offerts par Les APIs de la messagerie sans-fil (WMA) à travers les trois interfaces du package javax.wireless.messaging. Ces interfaces sont expliquées dans le tableau 5.1

| Interface | Description |
|---|---|
| *Message* | Cette interface représente un message. Les interfaces *TextMessage* et *BinaryMessage* sont dérivées à partir de Message |
| *MessageConnection* | Cette interface représente une connection réseau pour les messages. Elle définit les méthodes fondamentales pour envoyer et recevoir les messages. En l'occurrence, la méthode *MessageConnection.newMessage()* crée une instance d'un nouveau message sortant, alors que *MessageConnection.receive()* attrape, d'une manière asynchrone, un message entrant. |
| *MessageListener* | Cette interface a une seule méthode : *notifyIncomingMessage()*. Cette méthode est invoquée lorsqu'on détecte l'arrivée d'un message entrant |

TAB. 5.1 – Le package javax.wireless.messaging

La figure 5.3 montre le diagramme UML décrivant les relations entre les interfaces du package javax.wireless.messaging.

### 5.3.3 URLs et connections des messages

L'instanciation de la classe javax.microedition.io.Connector crée MessageConnection. La connection qu'on veut ouvrir est déterminée par l'URL passée à la méthode Connector.open(). WMA supporte les types de connections et d'URL suivants :

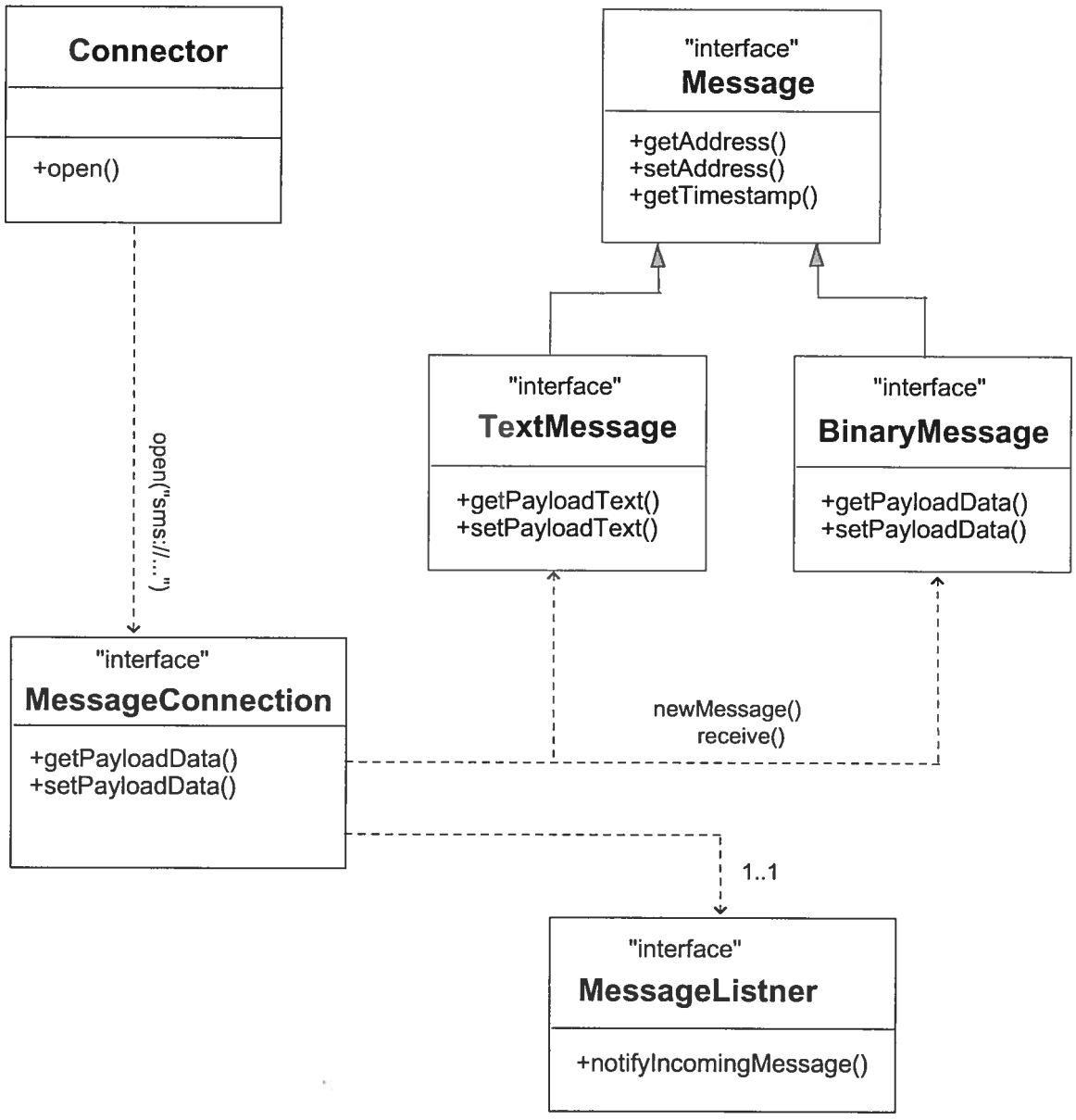



FIG. 5.3 – Interfaces WMA dans le package javax.wireless.messaging

- L'URL sms ://+18005555555 spécifie une connection pour envoyer des messages SMS au numéro de téléphone 1-800-555-5555.
- L'URL sms ://18005555555 :1234 spécifie une connection pour envoyer des messages SMS au port numéro 1234 sur le numéro de téléphone 1-800-555-5555.
- L'URL sms :// :1234 spécifie une connection de type serveur pour envoyer et recevoir des messages sur le pour 1234. Lors de l'envoie, on spécifie également le numéro de téléphone du destinataire.

Sans utilisation de port, un message reçu est intercepté par le système natif d'un téléphone. Aucune connection serveur de WMA n'est capable d'attraper de tels messages. Par contre, les paires de WMA échangent des messages SMS à travers des ports SMS privés pré-définis. Par exemple, lors de la réception d'un message SMS chiffré, on a intérêt à le recevoir par une application qui soit capable de le déchiffrer sur un port bien déterminé car le système natif est incapable de déchiffrer ces messages.

### 5.3.4 Envoi des messages SMS

On peut envoyer un message SMS à un numéro de téléphone mobile quelconque à travers la connection MessageConnection construite à partir de l'adresse URL du destinataire. Les codes des tableaux (5.2) et (5.3) montrent l'envoi des messages SMS.

```
String addr = "sms ://+123456789 :1234";
// ou String addr = "sms ://+123456789";
MessageConnection conn = (MessageConnection) Connector.open(adr);
TextMessage msg = (TextMessage) sconn.newMessage(
MessageConnection.TEXT_MESSAGE);
msg.setPayloadText("Bonjour SMS");
conn.send(msg);
```

TAB. 5.2 – Envoi d'un message SMS

```
String port = "sms :// :1234";
MessageConnection sconn = (MessageConnection) Connector.open(port);
TextMessage msg = (TextMessage) sconn.newMessage(
MessageConnection.TEXT_MESSAGE);
msg.setAddress("sms ://+123456789 :1234");
msg.setPayloadText("Bonjour SMS");
conn.send(msg);
```

TAB. 5.3 – Envoi d'un message SMS via une connection serveur

### 5.3.5 Reception des messages SMS

Dans une application J2ME, afin de recevoir des messages SMS, on a besoin d'avoir une MessageConnection serveur qui écoute l'arrivée des messages sur le port spécifié. Pour ce faire, il existe deux manière : synchrone et asynchrone. La méthode synchrone appelle la méthode MessageConenction.receive() itérativement pour traiter un message au moment de son arrivée. La méthode asynchrone, qui est beaucoup plus pratique, est basée sur l'utilisation des événements fournis par WMA. La méthode notifyIncomingMessage() de l'interface MessageListener est invoquée lorsqu'il y a détection d'une arrivée d'un message SMS.

### 5.3.6 Reception des messages via PUSH Registry

La méthode précédente de réception de messages SMS exige que l'application (la MIDlet), qui traite l'arrivé des messages, soit déjà démarrée. Dans le cas contraire, le message reçu, via un port spécifié, ne sera pas traité, ni par le système natif, ni par l'application car celle-ci n'est même pas démarrée. De ce fait, à partir de la version MIDP 2.0, on a un nouveau concept appelé PUSH Registry, qui permet à une MIDlet d'être démarrée automatiquement sur un appareil téléphonique lorsqu'un message SMS vient d'être reçu.

## 5.4 Conclusion

L'édition J2ME de Java avait été créée pour des appareils ayant une capacité de traitement et de stockage limitée, une connection en réseau (souvent sans fil), et d'une interface utilisateur graphique. Elle est unanimement adoptée sur la plupart des appareils téléphoniques mobiles, malgré leur diversité. En particulier, J2ME, combinée avec la configuration CLDC et le profil MIDP, fournit des APIs de la messagerie sans-fil (WMA) qui nous offrent la possibilité de développer notre propre MIDlet, appelée *TateSMS*, d'échange de messages SMS entre deux téléphones mobiles. Le prochain chapitre propose des mécanismes cryptographiques à apporter à TateSMS afin d'assurer la sécurité de ces messages.

# Chapitre 6

# Implémentation du système de chiffrement basé sur l'identité

L'objectif de ce chapitre est de décrire les algorithmes essentiels qui permettent de concrétiser le système de chiffrement FullIdent évoqué dans le chapitre 4. Les algorithmes sont présentés succinctement dans l'ordre prévu : Setup, Extract, Encrypt et Decrypt. Nous y trouverons, en particulier, l'algorithme de construction de la fonction de hachage $H_1$, qui transforme une identité (une chaîne de caractères) en un élément du groupe $\mathbb{G}_1$, et l'algorithme décrivant le calcul du couplage de Tate qui associe aux points $P, Q \in \mathbb{G}_1$ un élément $a \in \mathbb{F}_{p^2}$. Nous terminons ce chapitre par la présentation du diagramme de classes Java.

## 6.1 Algorithme *Setup*

L'algorithme *Setup* accepte en entrée un paramètre $k$, qui représente la taille en nombre de bits des groupes $\mathbb{G}_1, \mathbb{G}_2$, et génère les paramètres (publiques) du système $params =< q, \mathbb{G}_1, \mathbb{G}_2, \hat{t}, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$ et la clé maître $s$ qui demeure secrète. Cet algorithme est exécuté par le générateur de clés privées (PKG). Dans ce qui suit, nous allons détailler la manière dont chacun de ces paramètres est créé :

- $q$ : Un nombre premier de $k$ bits représentant l'ordre des groupes $\mathbb{G}_1, \mathbb{G}_2$.
- $\mathbb{G}_1$ : Le sous-groupe de $q$-torsion du groupe de la courbe elliptique supersingulière $E_{1,0}/\mathbb{F}_p$, où $p \equiv 3 \bmod 4$ est un nombre premier qui satisfait $q \mid (p^2 - 1)$ et $q \nmid (p - 1)$. Pour alléger l'écriture, $E$ désignera la courbe $E_{1,0}$.
- $\mathbb{G}_2$ : Le sous-groupe de $\mathbb{F}_{p^2}^*$ d'ordre $q$.
- $\hat{t} : \mathbb{G}_1^* \times \mathbb{G}_1^* \longrightarrow \mathbb{G}_2$ est le couplage de Tate réduit expliqué dans le chapitre 3. Ce couplage est implémenté dans l'algorithme 8 (section 6.3.2).
- $n$ : La taille du message en clair, en nombre de bits.
- $P$ : Est un point de la courbe $E/\mathbb{F}_p$ de $q$-torsion ($\in \mathbb{G}_1^*$) généré aléatoirement. Son algorithme de génération est expliqué dans la section 6.1.6.
- $P_{pub}$ : Est également un point du groupe $\mathbb{G}_1^*$ lié au point $P$ par la relation $P_{pub} = sP$, où $s$ est la clé maître.
- $H_1, H_2, H_3, H_4$ : Des fonctions de hachage du modèle de l'oracle aléatoire. Leurs algorithmes respectifs sont décrits dans les sous-sections suivantes. En plus, nous avons implémenté une fonction de hachage, nommée $H$, sur la base de la fonction de hachage standard $SHA1$, qui accepte en entrée une valeur de taille variable et retourne une valeur hachée de taille $n$.

### 6.1.1 La fonction de hachage $H$

Cette fonction de hachage est construite pour répondre aux exigences des deux fonctions de hachage $H_2$ et $H_4$ devant retourner des valeurs hachées de $n$ bits (la taille du message source). Nous avons ainsi décidé de construire $H$ à partir de la fonction de hachage standard $SHA1$ qui retourne une valeur hachée de 160 bits sur l'entrée

d'une chaîne de taille quelconque [Dou03].

Etant donnée $L$ une chaîne quelconque à hacher, le principe de la construction de $H$ est le suivant :

Soit $n = 160i + j$ avec $i > 0$ et $j < i$.

Mettre $L_k = "K"||L$ et

$H(L) := SHA1(L)||SHA1(L_0)||SHA1(L_1)||...SHA1(L_{i-1})_j \quad _{bits}$

La construction de la fonction $H$ est décrit dans l'algorithme 2. Ainsi, $H$ est bel et bien à sens unique, efficacement calculable et retourne une valeur aléatoire mais pas plus sécuritaire de la fonction SHA1 elle-même.

**Algorithme 2** Fonction de hachage H

**Entrées:** un entier $n \neq 0$ et une chaîne à hacher $L$
**Sorties:** une valeur hachée de $n$ bits : $\{0,1\}^n$
  **Si** $n > 160$ **Alors**
    $//n = 160i + j$
    $H(L) := SHA1(L)||SHA1("0"||L)||SHA1("1"||L)||...SHA1("i-1"||L)_j \quad _{bits}$
  **Sinon**
    $H(L) := SHA1(L)_n \quad _{bits}$
  **Fin Si**
  retourner $H(L)$

### 6.1.2 La fonction de hachage $H_1$

Rappelons que la fonction de hachage $H_1$ transforme une identité $ID \in \{0,1\}^*$ en un point $Q_{ID} \in \mathbb{G}_1^*$ d'ordre $q$. Afin de construire cette fonction, nous utilisons une idée de Boneh et Franklin [BF01] qui consiste à construire une fonction de hachage $H_{11}$ : $\{0,1\}^* \longrightarrow \mathbb{F}_p^*$ et une fonction de codage $H_{12} : \mathbb{F}_p^* \longrightarrow \mathbb{G}_1^*$. La fonction de hachage $H_{11}$ est construite à partir de SHA1 et transforme une chaîne de caractères quelconque

en une empreinte digitale de 160 bits. Celle-ci est ensuite convertie en un élément de $x_0 \in \mathbb{F}_P^*$. La fonction de codage $H_{12}$ recherche un point $Q_0 = (x_0, y_0) \in E/\mathbb{F}_P^*$ qui satisfait l'équation $y = x_0^3 + x_0$. Pour déterminer ce point, on calcule tout d'abord $y$ et on tente de trouver une racine carrée $y_0$ de $y$ dans $\mathbb{F}_p^*$. Cette racine existe si et seulement si $y$ est un résidu quadratique. Pour le test de résiduosité de $y$, on utilise le symbole de Legendre $\left(\frac{y}{p}\right) = y^{\frac{p-1}{2}}$. Ainsi, $y$ possède des racines si et seulement si le symbole de Legendre égal à 1. Dans le cas contraire, les racines sont inexistantes et on procède par incrémentation de $x_0$ jusqu'à ce que ce point soit trouvé.

Pour déterminer $Q_{ID}$ d'ordre $q$ à partir de $Q_0$, on multiplie ce dernier par $\frac{p+1}{q}$. Si $Q_{ID} = \mathcal{O}$, $x_0$ est attribué la valeur de l'incrément et tout l'algorithme est à recommencer jusqu'à ce que $Q_{ID}$ soit différent du point $\mathcal{O}$. L'algorithme 3 montre le fonctionnement de la fonction de hachage $H_1$.

### 6.1.3 La fonction de hachage $H_2$

Cette fonction transforme un élément de $\mathbb{G}_2^*$, qui est un élément de $\mathbb{F}_{p^2}$ d'ordre $q$ en une chaîne de $n$ bits. Rappelons qu'un élément de $\mathbb{F}_{p^2}$ est écrit sous la forme $a+bi$ où $a, b \in \mathbb{F}_p$ et $i^2 = -1$. Cette fonction procède comme suit : les chaînes binaires des deux composantes $a, b$ sont concaténées puis hachées par le biais de la fonction de hachage $H$ pour générer une valeur hachée de n bits.

### 6.1.4 La fonction de hachage $H_3$

Celle-ci accepte en entrée deux chaînes de $n$ bits chacune, et produit un élément de $\mathbb{Z}_q^*$. L'algorithme 4 montre son fonctionnement.

**Algorithme 3** Fonction de hachage $H_1$

**Entrées:** les points $ID \in \{0,1\}$
**Sorties:** $Q_{ID} \in \mathbb{G}_1^*$

$x_0 := (\mathbb{F}_p^*)SHA1(ID)$
$i := 1$; $more := true$
**Repeter**
  **Tant que** $(more)$ **Faire**
    $more = false$
    $y := x_0^3 + x_0$
    **Si** $\left(\frac{y}{p}\right) = 1$ **Alors**
      $y_0 := SQRT(y)$
    **Sinon**
      $x_0 := x_0 + 1$
      $more := true$
    **Fin Si**
  **Fin Tant que**
  $Q_0 := (x_0, y_0)$
  $Q_{ID} := \frac{p+1}{q} Q_0$
  **Si** $Q_{ID} = \mathcal{O}$ **Alors**
    $x_0 := i$
  **Fin Si**
  $i := i + 1$
**Jusqu'a** $Q_{ID} \neq \mathcal{O}$
retourner $Q_{ID}$

**Algorithme 4** Fonction de hachage $H_3$

**Entrées:** $\sigma$, $M \in \{0,1\}^n$
**Sorties:** $r \in \mathbb{Z}_q^*$

$result := H(\sigma || M)$
$r := (int)result \bmod p$
retourner $r$

### 6.1.5 La fonction de hachage $H_4$

Cette fonction transforme une chaîne de n bits en une chaîne aléatoire de même taille en nombre de bits ; $H_4 : \{0,1\}^n \longrightarrow \{0,1\}^n$. Elle est construite à partir de la fonction de hachage $H$.

### 6.1.6 Génération d'un point aléatoire de $q$-torsion

Le couplage de Tate utilisé dans le système de chiffrement IBE (voir le chapitre 3) requiert, en plus du point $Q_{ID}$ généré à partir d'une identité $ID$, un autre point $P \in G_1^*$ généré aléatoirement. Ce point génère le groupe $G_1$. En effet, son algorithme de génération diffère de celui de la fonction $H_1$ par le fait que $x_0$ est attribué une valeur initiale aléatoire au lieu d'une valeur hachée de l'identité $ID$.

### 6.1.7 Extraction d'une racine carrée dans $\mathbb{F}_p$

L'extraction des racines carrées dans $\mathbb{F}_p$ est une opération nécessaire durant les deux algorithmes *Setup* et *Extract* du système FullIdent. Pour extraire une racine carrée d'un élément $a \in \mathbb{F}_p$, on regarde tout d'abord si celui-ci est un résidu quadratique en examinant son symbole de Legendre. Si c'est le cas, les deux racines carrées sont $(r, -r)$ avec $r = a^{\frac{p+1}{4}} \bmod p$ lorsque $p \equiv 3 \bmod 4$. Pour plus de détails voir [MvOV97].

### 6.1.8 Opérations arithmétiques sur $E_{1,0}/\mathbb{F}_p$

Nous avons évoqué dans le premier chapitre les opérations arithmétiques effectuées sur les courbes elliptiques, soient l'addition et la multiplication par un scalaire. Dans l'implantation de notre système de chiffrement, ces opérations sont effectuées durant

les phases de génération des paramètres et d'extraction de clés privées (les algorithmes *Setup* et *Extract*) et aussi lors du calcul du couplage de Tate (*Encrypt* et *Decrypt*). L'algorithme 5 décrit la multiplication d'un point $P \in E_{1,0}/\mathbb{F}_p$ par un élément $m \in \mathbb{F}_p$ en faisant usage de la méthode *doubler et additionner*. Cet algorithme s'exécute en temps $O(log(m))$ . Rappelons que la multiplication par un scalaire dans le groupe $\mathbb{G}_1^* \subset E_{1,0}/\mathbb{F}_p$ est une loi interne, et tout élément $P \in \mathbb{G}_1^*$ est un générateur de celui-ci.

**Algorithme 5** Multiplication par un scalaire dans $E_{1,0}/\mathbb{F}_p$

**Entrées:** $P \in E_{1,0}/\mathbb{F}_p$ et $m \in \mathbb{F}_p^*$
**Sorties:** $Q = [m]P$
// soit $m = \sum_{i=0}^{n} m_i 2^i, \quad m_i \in \{0,1\}$ , $m_n = 1$
$Q := P$
**Pour** $i : n-1 \longrightarrow 0$ **Faire**
  $Q := Q + Q$
  **Si** $m_i = 1$ **Alors**
    $Q := Q + P$
  **Fin Si**
**Fin Pour**
retourner $Q$

## 6.2 Algorithme *Extract* : Génération d'une clé privée

Cet algorithme est exécuté, lui aussi, par le générateur de clés privées ($PKG$) sur réception d'une identité $ID$. Le PKG procède comme suit :

- Calcule $Q_{ID} = H_1(ID)$.
- Génère la clé privée $d_{ID} = sQ_{ID}$, où $s$ est la clé maître.

Dans le cas de la délégation des clés privées, déjà expliquée dans le chapitre 4, cette étape est sautée car la clé privée $d_{ID}$ est disponible chez l'usager ayant l'identité

$ID$ et qui demeure valide pendant une durée bien déterminée ; disons, le jour, le mois ou l'année en cours. Cette façon de procéder se prête bien pour l'implantation du système de chiffrement sur la messagerie cellulaire SMS, vu la difficulté d'extraction instantanée des clés privées au moment de la réception d'un message SMS crypté.

Dans le cas où la délégation des clés privées est effectuée pendant le mois courant, l'algorithme *Extract* est décrit ci-après.

**Algorithme 6** Algorithme *Extract*

**Entrées:** une identité $ID$
**Sorties:** la clé privée correspondante à $ID$ : $d_{ID}$
$Q_{ID} := H_1(ID||moisCourant())$
$d_{ID} := [s]Q_{ID}$
retourner $d_{ID}$

## 6.3 Algorithme de chiffrement : *Encrypt*

Nous disposons dès à présent de tous les éléments qui nous permettent d'implanter l'algorithme de chiffrement Encrypt. Reprenons les étapes de l'algorithmes de chiffrement FullIdent, telles quelles sont présentées dans le chapitre 4 :

Pour chiffrer un message $M \in \mathcal{M}$ en utilisant la clé publique $ID$ et les paramètres (publiques) du système $params =< q, \mathbb{G}_1, \mathbb{G}_2, \hat{t}, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$, l'algorithme fonctionne comme suit :

1. Calculer $Q_{ID} = H_1(ID) \in \mathbb{G}_1^*$.
2. Choisir un nombre aléatoire $\sigma \in \{0,1\}^n$.
3. Mettre $r = H_3(\sigma, M)$. Le cryptogramme est donné par :

$$C =< rP, \sigma \oplus H_2(g_{ID}^r), M \oplus H_4(\sigma) > \text{ où } g_{ID} = \hat{t}(Q_{ID}, P_{pub}) \in \mathbb{G}_2^*.$$

L'algorithme Encrypt est présenté dans l'algorithme 7

**Algorithme 7** Algorithme *Encrypt*

**Entrées:** $M \in \mathcal{M}$, $ID \in \{0,1\}^*$, avec $|M| = n$,
$params < q, \mathbb{G}_1, \mathbb{G}_2, \hat{t}, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$
**Sorties:** $C =< U, V, W > \in \mathcal{C}$, avec , $U \in \mathbb{G}_1^*, V, W \in \{0,1\}^n$
$Q_{ID} := H_1(ID||moisCourant())$
$\sigma := random\_string(n)$
$r := H_3(\sigma, M)$
$g_{ID} := FastTate(P_{pub}, \phi(Q_{ID}))$
$U := [r]P$
$V := \sigma \oplus H_2(g_{ID}^r)$
$W := M \oplus H_4(\sigma)$
retourner $C =< U, V, W >$

### 6.3.1 Opérations sur l'extension de corps $\mathbb{F}_{p^2}$

Nous avons évoqué dans le chapitre 2 qu'un élément du corps $F_{p^2}$ est représenté sous la forme $a + ib$ où $a, b \in \mathbb{F}_p$ et $i^2 = -1$. De ce fait, les opérations arithmétiques effectuées dans ce corps ressemblent à celles du corps des nombres complexes. Plus particulièrement, soit $x, y \in F_{p^2}$, avec $x = a + ib$ et $y = a' + ib'$. D'où, $x + y = (a + a') + i(b + b')$, $x.y = (a.a' - b.b') + i(a.b' + b.a')$ et $x^n$ est calculée successivement en $log(n)$ opérations à l'aide de l'algorithme *square and multiply.*

### 6.3.2 Calcul du couplage de Tate

Nous avons vu, au cours du chapitre 3, que le couplage de Tate de $P, Q \in \mathbb{G}_1$ est obtenu en une seule invocation de l'algorithme de Miller. Précisément, $\hat{t}(P, Q) = t(P, \phi(Q))^{\frac{p^2-1}{q}}$. Le point $P$ est généré aléatoirement, $Q$ est dérivé à partir d'une identité $ID$ connue publiquement, soit $Q = Q_{ID} = H_1(ID)$. L'algorithme de Miller permet, sur l'entrée de $P$ et $\phi(Q)$, de générer un élément $a \in \mathbb{G}_2^* \subset \mathbb{F}_{p^2}$, où $\phi$ est

l'application de distorsion $\phi : E/\mathbb{F}_p \longrightarrow E/\mathbb{F}_{p^2}, \phi(x,y) = (-x, iy)$. L'élévation à la puissance $\frac{p^2-1}{q}$ pourrait être simplifiée en tenant compte d'une idée vue dans la section 4.4.4 du chapitre 4.

La principale différence entre l'algorithme de Miller présenté dans le chapitre 4 et celui dans l'algorithme 8 réside dans le fait que ce dernier ne prend pas en compte les dénominateurs qui sont des facteurs non pertinents[1] pour le résultat final.

**Algorithme 8** Algorithme de calcul du couplage de Tate : *FastTate*

**Entrées:** $P \in E(\mathbb{F}_p)[q], Q \in E(\mathbb{F}_{p^2})[q]$ et un entier $q = \sum_{i=0}^{n} b_i 2^i$
**Sorties:** $t(P,Q) = f_P(Q) \in \mathbb{F}_{p^2}$

```
f := 1;   f_1 := 1
Z := O
Pour i : n − 1 ⟶ 0 Faire
  Si b_i = 1 Alors
    f := D(f, f_1, Z, P, Q)
    Z := Z + P
  Fin Si
  Si i > 0 Alors
    f := D(f, f, Z, Z, Q)
    Z := [2]Z
  Fin Si
Fin Pour
f := f^((p^2−1)/q)
retourner f
```

L'algorithme $D(V_1, V_2, P_1, P_2, Q)$ est une représentation de la fonction $f$ : $f_{a+b}(Q) = f_a(Q) \cdot f_b(Q) \cdot g_{aP,bP}(Q)/g_{(a+b)P}(Q)$ du lemme 3.1.1 dans le chapitre 3. Il retourne $V_1 \cdot V_2 \cdot ligne(P_1, P_2)$. L'algorithme *ligne*, décrit dans l'algorithme 9, évalue l'équation de la ligne qui passe à travers les deux points $P_1, P_2$ sur le point $Q$.

Pour obtenir la valeur du couplage, qui est un élément essentiel dans l'algorithme

[1] La non-pertinence des dénominateurs ($\in \mathbb{F}_p$) vient du fait que leurs valeurs respectives seront égales à 1 après l'élévation à la puissance $\frac{p^2-1}{q}$

**Algorithme 9** Evaluation de la ligne : *ligne*

**Entrées:** $P_1, P_2 \in E(\mathbb{F}_p), Q \in E(\mathbb{F}_{p^2}), V_1, V_2 \in \mathbb{F}_{p^2}$
**Sorties:** $ligne(P_1, P_2) \in \mathbb{F}_{p^2}$ : évaluation de la ligne à travers $P_1(x_1, y_1)$ et $P_2(x_2, y_2)$ sur le point $Q(x_Q, y_Q)$

$l := 1$
**Si** $P_1 = \mathcal{O}$ **Alors**
  retourner $l$
**Sinon Si** $P_1 \neq P_2$ **Alors**
  $slope := \frac{y_2 - y_1}{x_2 - x_1}$
  $l := y_Q - y_1 - slope(x_Q - x_1)$
**Sinon**
  // la tangente au point $P_1$
  // $a = 1$ *pour* $E_{1,0}$
  $slope := \frac{3x_1^2 + a}{2y_1}$
  $l := y_Q - y_1 - slope(x_Q - x_1)$
**Fin Si**
retourner $l$

de chiffrement, la valeur $f$, retournée par l'algorithme de Miller, est élevée à la puissance $\frac{p^2-1}{q}$.

## 6.4 Algorithme de déchiffrement : *Decrypt*

L'algorithme *Decrypt* effectue l'opération inverse de l'algorithme *Encrypt*, sauf que cette fois-ci, on aura besoin de l'intervention de l'autorité de génération des clés privées (*PKG*). *Decrypt* accepte en entrée un cryptogramme $C =< U, V, W >$, généré à l'aide de l'algorithme *Encrypt* en utilisant l'identité $ID$, et la clé privée $d_{ID}$ correspondant à cette identité [2] et, après vérification de l'integrité du texte chiffré, produit le texte clair $M$ correspondant au cryptogramme $C$. En voici les étapes de l'algorithme *Decrypt* :

1. Calculer $V \oplus H_2(\hat{t}(d_{ID}, U)) = \sigma$.

[2] $d_{ID} = s \cdot Q_{ID}$, où $s$ est la clé maître connue seulement par le (*PKG*)

2. Calculer $W \oplus H_4(\sigma) = M$.

3. Mettre $r = H_3(\sigma, M)$. Vérifier si $U = rP$. Sinon, refuser le texte chiffré.

4. Produire $M$ comme étant le déchiffrement de $C$.

**Algorithme 10** Algorithme *Decrypt*

**Entrées:** $C =< U, V, W > \in \mathcal{C}$, $d_{ID} \in \mathbb{G}_1^*$,
$params =< q, \mathbb{G}_1, \mathbb{G}_2, \hat{t}, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$
**Sorties:** le texte clair $M$, ou erreur d'intégrité
$\sigma := V \oplus H_2(\hat{t}(d_{ID}, U))$
$M := W \oplus H_4(\sigma)$
$r := H_3(\sigma, M)$
**Si** $U \neq [r]P$ **Alors**
*Erreur d'integrité*
**Fin Si**
retourner $M$

## 6.5 Diagramme de classes

La concrétisation des algorithmes précédents conduit à l'élaboration des principales classes Java suivantes :

**TateSMS :** est la principale classe du projet avec comme attributs port : le port par lequel transitent les messages SMS entrant et sortant, Message : le message à envoyer, k : la taille, en nombre de bits, des groupes $\mathbb{G}_1, \mathbb{G}_2$ à créer. Parmi ses méthodes, on trouve SendMessage(String Num_Tel, String message) chargée d'envoyer le chiffrement du message désigné au numéro de téléphone Num_Tel. Pour cela, elle fait appel à la méthode de chiffrement Encrypt(SParams params, String TexteClair, ECCPoint Qid) qui implémente l'algorithme 7 et retourne le cryptogramme associé au TexteClair. Par ailleurs, la méthode notifyIncomingMessage()

implémente un écouteur (*listener*) de messages (chiffrés) reçus sur le port port et déclenche, à cet effet, un événement qui instancie la classe ReceiveMessage().

**ReceiveMessage :** Cette classe fait appel à la méthode Decrypt(SParams params, String CipherText, ECCPoint $d_{id}$), où $d_{id}$ est la clé privée associée à l'identité $ID$ qui représente le numéro de téléphone du destinataire.

**SParams :** implémente l'algorithme *Setup*. Elle prend en entrée l'attribut $k$ et produit tous les paramètres du système $params =< q, \mathbb{G}_1, \mathbb{G}_2, \hat{t}, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$ évoqués précédemment. Rappelons que cet algorithme est exécuté par l'autorité de génération des clés privées.

**FastTate** implémente l'algorithme de calcul du couplage de Tate réduit (voir l'algorithme 8). Celle-ci est instanciée deux fois ; au moment du chiffrement :

FastTate t = new FastTate($q, Ppub, Q_{id}, courbe$)

et lors du déchiffrement :

FastTate t = new FastTate($q, d_{ID}, U, courbe$)

D'autres classes sont utilisées, comme ECCurve(p, a, b) [3] et ECCPoint(ECCurve courbe, $P_x$, $P_y$) qui implémentent les courbes elliptiques et les opérations arithmétiques associées. Celles-ci sont inspirées, en partie, de celles fournies par l'organisation BouncyCastle [4].

[3] $p$ est la caratéristique du corps $\mathbb{F}_p$ , $a, b$ les coeffecients de la courbe elliptique $E_{a,b}$

[4] www.bouncycastle.org

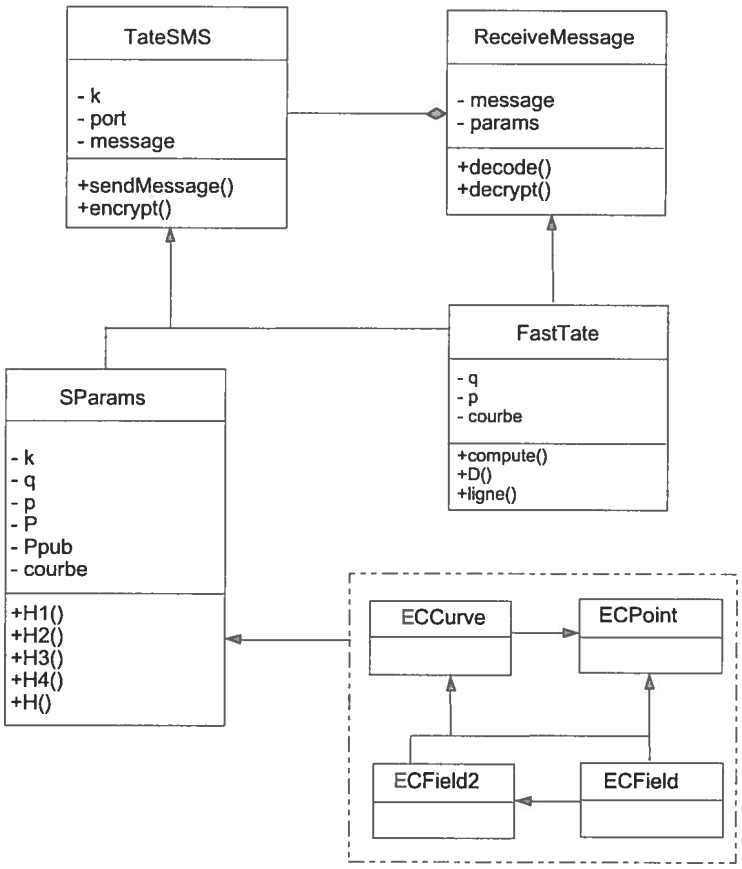



FIG. 6.1 – Le diagramme de classes

## 6.6 Exemple d'exécution et de fonctionnement

Pour montrer un exemple d'utilisation du système de chiffrement implanté, supposons que Alice et Bob décident, cette fois-ci, d'échanger des messages SMS confidentiels sur leurs téléphones mobiles. Ils décident ainsi d'utiliser le système FullIdent installé dans leurs cellulaires sous forme d'une application (MIDlet) appelée *TateSMS*. Leur admiration pour ce système provient du fait qu'on n'exige aucun mot de passe, ni pendant le chiffrement, ni à la réception d'un message chiffré.
Montrons le déroulement des quatre algorithmes fondamentaux du système de chiffrement, soient Setup, Extract, Encrypt et Decrypt.

**L'algorithme Setup :** Cet algorithme est exécuté par le générateur de clés privées (PKG), sur l'entrée d'un paramètre $k$, égal à 97 dans cet exemple. On génère ainsi les paramètres publiques du système $params =< q, \mathbb{G}_1, \mathbb{G}_2, \hat{t}, n, P, P_{pub}, H_1, H_2, H_3, H_4 >$ comme montré dans la section (6.1). Notamment, $q$ est un nombre premier de $k$ bits représentant l'ordre des groupes $\mathbb{G}_1$ et $\mathbb{G}_2$, $\hat{t}$ est le couplage de Tate réduit. Cet algorithme génère également la clé maître $s \in \mathbb{Z}_q^*$ gardée secrètement par le PKG.

**L'algorithme Extract :** Le PKG procède à la génération de la clé privée de Bob $d_{ID}$ pour que ce dernier puisse déchiffrer les messages SMS reçus. Cette clé est dérivée à partir de l'identité (le numéro de téléphone) de Bob et de la clé maître $s$. En adoptant la notion de délégation des clés privées [BF01], $d_{ID}$ serait valide uniquement pendant une période bien déterminée, par exemple le mois en cours (eg. 1204). Le PKG effectue donc les opérations suivantes :

- Calcule $Q_{ID} = H_1(ID||"1204")$
- Génère la clé privée $d_{ID} = sQ_{ID}$

Les paramètres publiques *params* et la clé privée associée à l'identité de Bob feront désormais partie de la MIDlet TateSMS déployée sur le téléphone mobile de Bob.

**L'algorithme Encrypt :** Lorsque Alice décide d'envoyer un message chiffré à Bob, tout ce qu'elle a besoin d'introduire est le numéro de téléphone ($ID$) de Bob et le message SMS source $M$, de taille $n$ bits, à lui envoyer. Pour générer le cryptogramme, le système ayant les paramètres publiques *params*, procède comme suit :

- Calculer $Q_{ID} = H_1(ID||moisCourant())$
- Générer une chaîne aléatoire $\sigma \in \{0,1\}^n$
- Mettre $r = H_3(\sigma, M) \in \mathbb{Z}_q^*$
- Envoyer le cryptogramme $C$ à Bob sous forme d'un message SMS, où $C =< rP, \sigma \oplus H_2(g_{ID}^r), M \oplus H_4(\sigma) >$ avec $g_{ID} = \hat{t}(Q_{ID}, P_{pub}) \in \mathbb{G}_2^*$

La fonction moisCourant() retourne le mois en cours sous la forme "MMAA" (eg. "1204").

**L'algorithme Decrypt :** En utilisant les paramètres publiques *params* et la clé privée $d_{ID}$ (valide) de Bob, le système serait capable d'effectuer l'opération de déchiffrement du cryptogramme reçu. Pour générer le message SMS source, le système procède comme indiqué dans l'algorithme 10 de ce chapitre.

Nous présentons dans l'annexe A les instructions Java à exécuter et les résultats obtenus pour chaque algorithme cité plus haut.

La figure 6.2 montre l'exécution de la MIDlet TateSMS sur un émulateur de cellulaire de *J2ME Wireless Toolkit.*

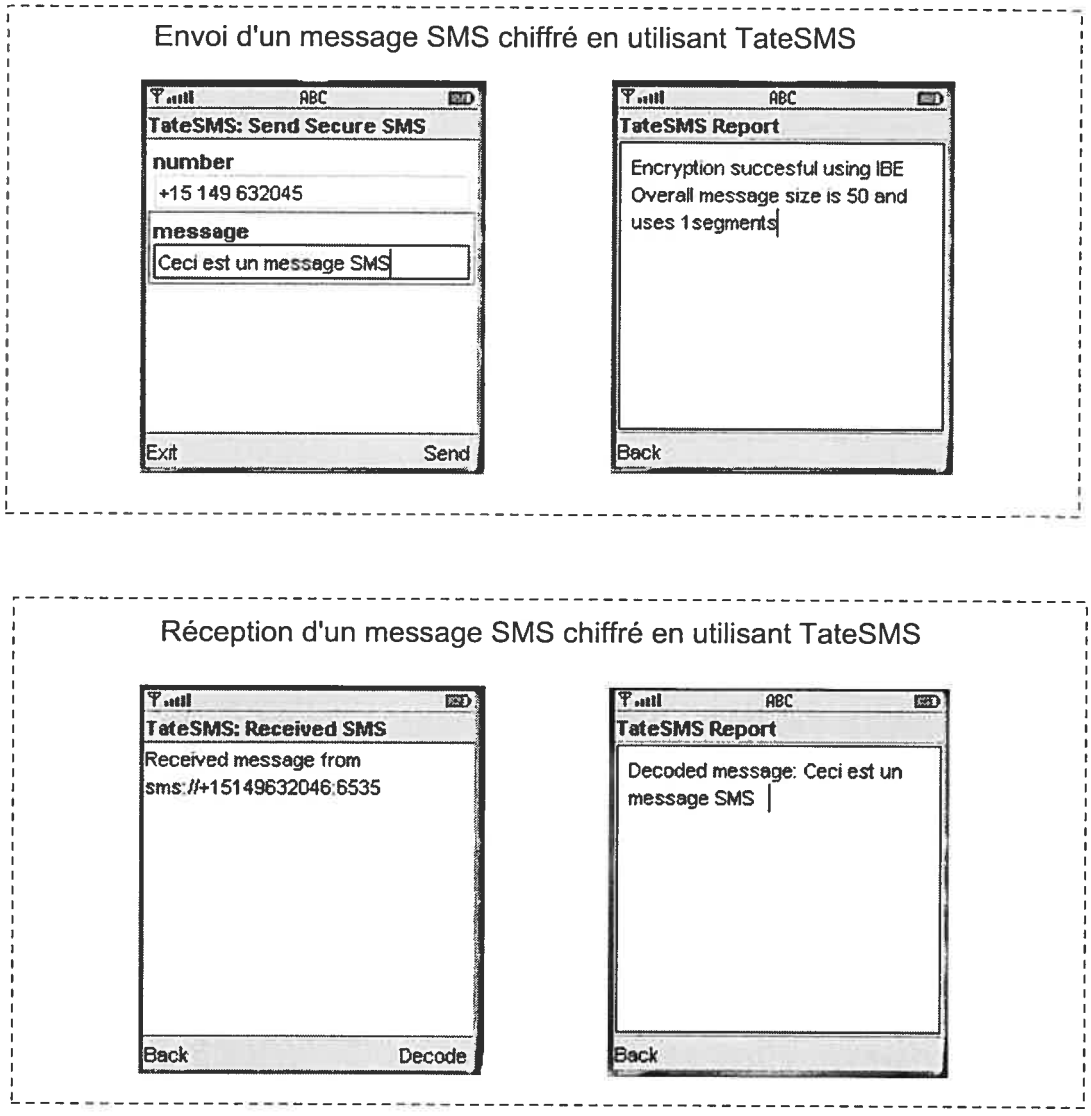



FIG. 6.2 – Fonctionnement de TateSMS

# Chapitre 7

# Analyse de performances

Pour une MIDlet, les deux préoccupations majeures sont son délai d'exécution et sa taille en kilo octet. Si la taille maximale d'une MIDlet est spécifiée au sein de la configuration CLDC, le délai d'exécution doit rester plutôt raisonnable surtout pour une application interactive.

Les performances de notre système se focalisent sur les facteurs suivants :

- La vitesse d'exécution, qui est étroitement liée au choix des deux paramètres $q$ et $p$.
- La sécurité du système, qui lui aussi dépend des tailles des deux nombres premiers $q$ et $p$.
- Le nombre de caractères pouvant être envoyés/reçus au sein d'un message SMS.

Nous étudions les différents scénarios qui permettent de maximiser le nombre de caractères dans un message SMS tout en assurant une sécurité acceptable pour celui-ci. Les tests effectués sont relatifs à un émulateur de cellulaire Nokia 6230 et peuvent, à cet effet, changer sur un autre modèle d'appareil téléphonique mobile.

## 7.1 Structure du cryptogramme

Le cryptogramme généré par le système FullIdent, à partir d'un message source $M$ de taille $n$ caractères, est un triplet $C =< U, V, W >$ où $U \in \mathbb{G}_1$ et $V, W$ sont des chaînes de $n$ caractères chacune. En outre, la trame envoyée (le cryptogramme) est accompagnée du nombre entier $n$. On réserve deux octets pour représenter $n$ afin de pouvoir envoyer, quand cela est possible, un cryptogramme dont le nombre de caractères dépasse 256. La structure d'un cryptogramme à envoyer sous la forme d'un message SMS est décrit sur la figure suivante :

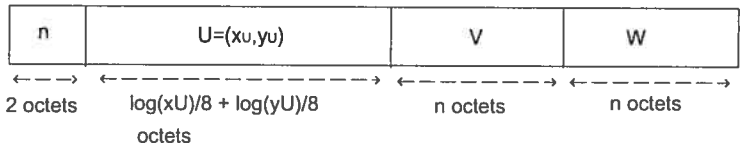



FIG. 7.1 – La structure d'un cryptogramme

Sachant que le nombre maximal de caractères pouvant être envoyés au sein d'un message SMS est souvent limité, disons à $L$ caractères, nous pouvons en déduire la relation suivante :

$$2 + 2n + \frac{log(x_U) + log(y_U)}{8} \leq L$$

Avec $x_U, y_U \in \mathbb{F}_p$, on a $log(x_U) + log(y_U) \leq 2log(p)$, d'où $2 + 2n \leq L - \frac{log(p)}{4}$

Afin de garantir l'envoi de $n$ caractères sans éventuelle perte de données, nous avons :

$$n_{max} = \frac{4L - log(p) - 8}{8} \tag{7.1.1}$$

Par exemple, pour un premier $p$ de 256 bits et $L = 160$ caractères, nous avons $n_{max} = 47$ caractères.

Par conséquent, la taille de $p$, en nombre de bits, est un choix déterminant pour le nombre maximal de caractères pouvant être envoyés dans un message SMS.

## 7.2 Compression du cryptogramme

Par souci d'optimisation, nous proposons une méthode qui permet de réduire la taille du cryptogramme, étant donnée la contrainte liée à la taille maximale d'un message SMS. Pour cela, nous optons pour la compression du point $U = (x_U, y_U)$ à l'envoi du cryptogramme et sa décompression à la réception. Cette compression consiste à envoyer seulement la composante $x_U$ de $U$, en y ajoutant une information supplémentaire $B$, représentée sur un octet, permettant de déterminer la composante $y_U$ tel que $y_U^2 = x_U^3 + x_U$ lors de l'opération de décompression. Pour cela, on sauvegarde dans $B$ le bit le moins significatif de $y_U$. La structure du cryptogramme compressé est montrée sur la figure 7.2

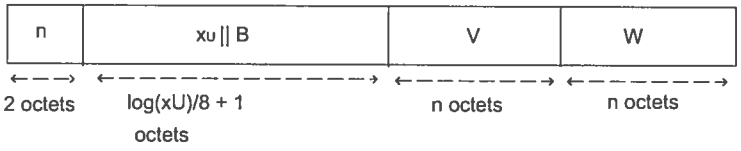



FIG. 7.2 – La structure d'un cryptogramme compressé

Ainsi, nous obtenons une nouvelle formule reliant le nombre maximal de caractères dans un message SMS à la taille, en nombre de bits, du nombre premier $p$. Soit : $2 + \frac{log(x_U)}{8} + 1 + 2n = L$

ou

$$n = \frac{8L - log(x_U) - 24}{16}$$

Par conséquent, le nombre maximal de caractères pouvant être envoyés dans un message SMS sans risque de perte de données est :

$$n_{max} = \frac{8L - log(p) - 24}{16} \tag{7.2.1}$$

Par exemple, pour $L = 160$ caractères, et $p$ de taille 256 bits, nous avons $n_{max} = 62$ caractères.

## 7.3 Analyse de la sécurité du système

La sécurité du système de chiffrement proposé repose sur la difficulté du problème de Diffie-Hellman Bilinéaine (BDH)[1] présenté dans le chapitre 4. En effet, aucune solution n'est actuellement connue à ce problème, apart sa réduction en un problème de logarithme discret traditionnel. Rappelons que le BDH est exprimé comme suit : Étant donnés $P, aP, bP, cP \in \mathbb{G}_1$ avec $a, b, c \in \mathbb{Z}_q^*$, et un couplage bilinéaire $t$ non dégénéré, il est difficile de calculer $t(P, P)^{abc}$.

Dans le cas du système proposé, nous disposons des paramètres publiques suivants : $P, P_{pub}, U, Q_{ID} \in \mathbb{G}_1$, avec[2] $P$ un générateur dans $\mathbb{G}_1$. Respectivement, ces paramètres peuvent être exprimés par $P, sP, rP, xP$, avec $s, r, x \in \mathbb{Z}_q^*$ des paramètres inconnus. Le problème du BDH recherche à calculer $t(P, P)^{srx} = t(d_{ID}, U) = t(Q_{ID}, P_{pub})^r = g_{ID}^r$.

[1] Bilinear Diffie-Hellman

[2] $U$ fait partie du cryptogramme, et $Q_{ID} = H_1(ID)$ est publique car $H_1$ et $ID$ sont publiques

Notamment, cette valeur nous permet de déchiffrer un cryptogramme sans même connaître la valeur de la clé privée $d_{ID}$.

Étant donnée l'hypothèse de difficulté du DBH, nous explorons une autre piste liée à un problème traditionnel ; il s'agit du logarithme discret.

### 7.3.1 Logarithme discret dans le groupe $\mathbb{G}_1$

Étant donnés les deux paramètres publiques $P$ et $P_{pub} = sP$, où $s$ est la clé maître secrète, appartenant au groupe $\mathbb{G}_1$. Une attaque contre le logarithme discret sur les courbes elliptiques (ECDLP[3]) vise à déterminer $s \in Z_q^*$ à partir de $P$ et $sP$. Cependant, la force cryptographique du chiffrement basé sur les courbes elliptiques découle de la difficulté pour une cryptanalyse de déterminer le secret $s$. En effet, le meilleur algorithme pour résoudre ce problème est *Rho* de Pollard qui s'exécute en un temps exponentiel de l'ordre de $O(\sqrt{q})$ et nécessite très peu d'espace mémoire [MvOV97] [LV01]. Par ailleurs, Certicom[4] montre dans [Cer] le nombre d'opérations nécessaires et le délai global pour pouvoir calculer le logarithme discret dans un groupe de points d'une courbe elliptique. Par exemple, le calcul du logarithme discret sur une courbe elliptique d'ordre premier $q$ de 131 bits demande $3,5 \times 10^{19}$ opérations dans le groupe. Soit environ $8,55 \times 10^6$ jours de calcul sur un ordinateur Pentium IV 2,66 GHZ.

Le tableau 7.1 montre l'estimation de la durée à consommer pour pouvoir calculer le logarithme discret dans un groupe de points d'une courbe elliptique d'ordre $q$ variable.

[3]ECDLP : Elliptic Curve Discrete Logarithm Problem
[4]www.certicom.com

| $log(q)$ | Nombre d'opérations dans le groupe | Nombre d'itération/sec. | Nombre de Jours de calcul |
|---|---|---|---|
| 89 | $1,8 \times 10^{13}$ | 1276800 | 163 |
| 97 | $3,14 \times 10^{14}$ | 1276800 | 2719 |
| 109 | $2,1 \times 10^{16}$ | 718200 | 338423 |
| 131 | $3,5 \times 10^{19}$ | 474012 | $8,55 \times 10^{6}$ |
| 163 | $2,4 \times 10^{24}$ | 319200 | $8,7 \times 10^{13}$ |

TAB. 7.1 – Estimation du temps de calcul du ECDLP en utilisant la méthode *Rho* de Pollard sur un Pentium IV 2,66 GHz

Lorsque la courbe elliptique est supersingulière, le problème de logarithme discret dans le groupe $\mathbb{G}_1$ est beaucoup moins facile à calculer. Ceci est dû à la réduction de MOV, en utilisant le couplage de Weil, ou FR en utilisant le couplage de Tate (voir chapite 4). À l'aide de cette réduction, le problème du logarithme discret dans le groupe $\mathbb{G}_1$ est ramené au même problème dans le groupe $\mathbb{G}_2$.

### 7.3.2 Logarithme discret dans le groupe $\mathbb{G}_2$

Le calcul du logarithme discret dans le groupe $\mathbb{G}_2 \subset \mathbb{F}_{p^2}$ est beaucoup moins facile. La raison en est qu'il existe, dans ce groupe, des algorithmes qui permettent de calculer efficacement ce problème en un temps sous-exponentiel. Parmi ces algorithmes, on cite Index Calculus Method (ICM) et Number Field Sieve (NFS) avec un temps d'exécution de l'ordre de $L_{p^2}[\frac{1}{2}, \frac{3}{2}]$ [MvOV97], [Odl00].

Soit $N_1 = \lceil \log_2 q \rceil$ et $N_2 = \lceil \log_2 p^2 \rceil$.

Pour que les deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ aient des sécurités similaires, [IGN99] propose

une dérivation de $N_1$ à partir de $N_2$ :

$$N_1 \approx 4.91 N_2^{1/3} (\ln(N_2 \ln 2))^{2/3} \tag{7.3.1}$$

Par conséquent, La sécurité du système de chiffrement basé sur l'identité dépend à la fois de $N_1$ et $N_2$ suivant la relation 7.3.1. Par exemple, Les tailles des clés $q$ de 109, 131 et 163 bits dans le groupe $\mathbb{G}_1$ doivent correspondre à des valeurs de $p$ de 140, 271 et 444 bits respectivement dans $\mathbb{G}_2$. Autrement dit, des tailles de l'ordre de 280, 245 et 888 bits, étant donnée que la taille du groupe $\mathbb{G}_2 \subset \mathbb{F}_{p^2}$ égale à $2p$.

## 7.4 Impact de $q$ sur l'efficacité du système

Un regard sur l'algorithme de calcul du couplage de Tate nous permet de déterminer qu'il y a une corrélation certaine entre la taille et la nature du nombre $q$ et les délais de chiffrement/déchiffrement. Rappelons que $q$ est l'ordre des groupes $\mathbb{G}_1$ et $\mathbb{G}_2$. En effet, nous remarquons qu'il y a moins d'opérations lorsque le nombre d'éléments binaires dans $q$ est minimal et vice-versa. Barreto et al. [BLS03] proposent l'utilisation des nombres premiers pseudo-Mersenne, appelés aussi de type Solinas [Sol99]. Ces nombres premiers sont de la forme $2^x \pm 2^y \pm 1$ avec $x > y > 0$. Le tableau 7.2 montre l'amélioration du délai de chiffrement pour un $q$ de taille 98 bits de type Solinas, et un autre $q$ aléatoire de même taille. Dans les deux cas, $p$ est un nombre premier aléatoire de taille 140 bits.

| $\|q\| = 98bits$ | Délai de chiffrement en ms |
|---|---|
| aléatoire | 6064 |
| $2^{97} + 2^{16} + 1$ | 4895 |

TAB. 7.2 – Influence des nombres $q$ Solinas sur le délai de chiffrement

## 7.5 Sauvegarde de la valeur du couplage

Une autre idée pour améliorer davantage les délais de chiffrement consiste à sauvegarder la valeur du couplage associée à une identité $ID$ donnée[5]. En effet, nous savons que, sous les mêmes paramètres du système $params$, une identité $ID$ détermine de façon unique la valeur intermédiaire du couplage associé. Plus précisément, la valeur $g_{ID} = t(P_{pub}, Q_{ID})$ peut être sauvegardée dans la mémoire du cellulaire aussi bien que l'identité $ID$ elle-même. Cette dernière jouera le rôle de la clé de recherche.

En faisant ainsi, lors d'un prochain chiffrement avec la même identité $ID$, la valeur $g_{ID}$ est tout simplement restaurée de la mémoire au lieu d'être recalculée. Ceci épargne une bonne partie du temps de chiffrement consommé pendant cette opération.

Le tableau 7.3 montre le gain en délai de chiffrement obtenu après implantation de cette méthode.

Avec $q$ un nombre premier de type Solinas, et $p = 3 \bmod 4$ lui aussi un nombre premier tel que $q \mid p^2 - 1$ et $q \nmid p - 1$. En plus, $q$ et $p$ sont choisis de telle sorte que les deux groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ aient des sécurités équivalentes selon la relation 7.3.1.

Intuitivement, cette méthode est pratique lorsque le nombre d'identités manipulées est raisonnable.

[5]Ici ID représente un numéro de téléphone

| $\|q\|$ | $\|p\|$ | délai de chiffrement avant sauvegarde de $g_{ID}$ en ms | délai de chiffrement après sauvegarde de $g_{ID}$ en ms |
|---|---|---|---|
| 89 | 116 | 3775 | 2519 |
| 98 | 140 | 4895 | 3278 |
| 109 | 180 | 7510 | 5061 |
| 131 | 271 | 18643 | 11480 |
| 160 | 426 | 49737 | 30921 |

TAB. 7.3 – Impact de la sauvegarde de $g_{ID}$ sur le délai de chiffrement

Quant à la réception, la valeur du couplage à calculer est $t(U, d_{ID}) = t(rP, d_{ID}) = g_{ID}^r$ qui dépend, en plus de l'identité $ID$, d'une valeur $r \in \mathbb{Z}_q^*$ choisie aléatoirement à l'envoi. Cependant, il est inutile de se servir de la valeur $g_{ID}$ car, en tout cas, la valeur de $r$ reste inconnue. Ceci dit, la sauvegarde des valeurs de $(ID, g_{ID})$ améliore certes le délai de chiffrement mais non pas celui du déchiffrement.

# Chapitre 8

# Conclusion et travaux futurs

## 8.1 Conclusion

Dans ce mémoire, nous avons montré que l'insécurité de SMS nous incite à réfléchir sur une solution adéquatement sécuritaire et simple à utiliser. Dès lors, nous avons passé en revue les différents modes de chiffrement existants, pour enfin choisir le chiffrement basé sur l'identité (IBE) pour ses avantages intangibles. Un système IBE utilise l'identité du destinataire (dans notre cas, son numéro de téléphone mobile) comme clé publique de chiffrement. De ce fait, on n'a pas besoin de la certification des clés publiques, ni d'une autorité de certification à l'envoi. En outre, ce mode de chiffrement incarne la possibilité de révocation des clés privées et est parfaitement extensible.

Les couplages bilinéaires représentent la pierre angulaire dans la construction du système de chiffrement proposé. À cet effet, nous avons développé, à travers les premiers chapitres, la théorie nécessaire qui mène à la construction du couplage de Tate réduit, dont le calcul comporte le moins d'opérations arithmétiques possibles comparativement à d'autres couplages.

La sécurité de notre système repose sur la difficulté du logarithme discret, à la

fois sur le groupe $\mathbb{G}_1$ et sur $\mathbb{G}_2$. Plus précisément, sur le groupe le moins faible. Toutefois, un compromis doit se faire entre la sécurité et l'efficacité qui sont, en principe, diamétralement opposées.

Le déploiement de l'application J2ME (la MIDlet) TateSMS que nous avons développé suppose l'existence d'un canal sécuritaire entre l'autorité PKG et le téléphone mobile d'un usager. Pour cela, on suggère, à titre d'exemple, de télécharger la MIDlet, à travers une connection HTTPS, sur un ordinateur local, ensuite de l'installer sur l'appareil mobile. En principe, cette opération s'effectue à la toute première installation et à chaque renouvellement d'une clé privée expirée.

## 8.2 Travaux futurs

Bien que la MIDlet TateSMS fonctionne d'appareil mobile à appareil mobile, une solution de type client-serveur ou appareil mobile à PC est une piste envisageable. Dans ce cas, les délais de traitement (chiffrement et déchiffrement) seront plus ou moins négligeables sur le serveur.

La version actuelle de TateSMS ne permet pas la diffusion simultanée d'un message SMS vers plusieurs destinataires. Cette solution pourrait être explorée en ayant recours aux couplages multi-linéaires [BF01]. Par contre, la diffusion ou l'envoi séquentiel d'un SMS vers plusieurs destinataires est simple à implanter mais le délai de chiffrement serait linéaire dans le nombre de destinataires.

Le système implanté pourrait facilement être incorporé dans un système de chiffrement de courriels basé sur l'identité. Dans ce cas, il convient de prendre l'adresse de courriel du destinataire, et possiblement concaténée avec d'autres informations comme la ligne d'objet, comme étant une identité de chiffrement. Ceci facilitera la

délégation des tâches des employés comme expliqué dans le chapitre 4. Seuls les employés concernés par une ligne d'objet déterminée pourront alors consulter les courriels qui leur sont destinés.

Par souci de rapidité de l'arithmétique des points des courbes elliptiques, et donc l'accélération des délais de chiffrement et déchiffrement, il a été conclu dans [IT03] et [CMO98] que l'utilisation d'une autre représentation des coordonnées de points des courbes elliptiques, telle que la représentation *projective* ou *jacobienne*, accélère d'une manière significative les délais de calcul des opérations arithmétiques dans ces courbes, et donc aussi bien du couplage de Tate. Ceci est dû principalement à l'opération de calcul de l'inverse dans le corps $\mathbb{F}_p$ nécessaire dans la représentation affine.

# Bibliographie


[BF01] D. Boneh et M. Franklin. « Identity-Based Encryption from the Weil Pairing ». *Advances in Cryptology - Crypto 2001, LNCS 2139, Springer-Verlag*, pp. 213–229, 2001.

[BKLS02] P. Barreto, H. Kim, B. Lynn, et M. Scott. « Efficient Algorithms for Pairing-Based Cryptosystems ». *dvances in Cryptology - Crypto 2002, LNCS , Springer Verlag*, vol. 2442 :pp. 354–368, 2002.

[BLS03] P. Barreto, B. Lynn, et M. Scott. « On the Selection of Pairing-Friendly Groups ». *Cryptology ePrint Archive, Rapport 2003/086*, 2003. Disponible au http ://eprint.iacr.org/2003/086.

[BW03] F. Brezing et A. Weng. « Elliptic curves suitable for pairing based cryptography ». *Cryptology ePrint Archive*, (2003/143), 2003. Disponible au : http ://www.cs.up.ac.za/cs/jbishop/Homepage/Pubs/Tech-reports/SecureSMS.pdf.

[Cer] Certicom. « The Certicom ECC Challenge ». Disponible au http ://www.certicom.com/index.php ?action=res,ecc_challenge.

[CL02] Jung Hee Cheon et Dong Hoon Lee. « Diffie-Hellman Problems and Bilinear Maps ». *Cryptology ePrint Archive, Rapport 2002/117*, 2002. Disponible au http ://eprint.iacr.org/2002/117.

[Cle02] T. Clements. « SMS–Short but Sweet ». *Sun Microsystems*, 2002. Disponible au : http ://developers.sun.com/techtopics/mobility/midp/articles/sms/.

[CMO98] H. Cohen, A. Miyaji, et T. Ono. « Efficient Elliptic Curve Exponentiation Using Mixed Coordinates ». *ASIACRYPT : Advances in Cryptology : International Conference on the Theory and Application of Cryptology, LNCS, Springer-Verlag*, 1998.

[Coc01] C. Cocks. « An Identity Based Encryption Scheme based on Quadratic Residues ». *Proc. of 8th IMA International Conference on Cryptography and Coding, LNCS 22260, Springer Verlag*, pp. 360–363, 2001.

[Dou03] R. Stinson Douglas. « *Cryptographie : théorie et pratique* ». traduction de Serge Vaudenay, Gildas Avoine et Pascal Junod ; Deuxième édition, Vuibert informatique, Paris, 2003.

[Eng99] Andreas Enge. « *Elliptic Curves and Their Applications to Cryptography - An Introduction* ». Kluwer Academic, 1999.

[FMR99] G. Frey, M. Müler, et H. Rück. « The Tate pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystem ». *IEEE Transactions on Information Theory*, vol. 45(5) :pp. 1717–1719, Juillet 1999.

[FO99] E. Fujisaki et T. Okamoto. « Secure integration of asymmetric and symmetric encryption schemes ». *Advances in Cryptology - Crypto '99 , Lecture Notes in Computer Science*, vol. 1666 :pp. 537–554, 1999.

[FOR03] NOKIA FORUM. « A Brief Introduction to Secure SMS Messaging in MIDP ». 2003. Disponible au http ://ncsp.forum.nokia.com/download/ ?asset_id=453 ;ref=devx.

[IGN99] Blake I.F., Seroussi G., et Smart N.P.. « *Elliptic curves in cryptography* ». Cambridge University Press, 1999.

[IT03] Tetsuya Izu et Tsuyoshi Takagi. « Efficient Computations of the Tate Pairing for the large Mov degrees ». *ICISC : International Conference on the Theory and Application of Cryptology, LNCS, Springer-Verlag*, vol. 2587 :pp. 283–297, 2003.

[JN01] A. Joux et K. Nguyen. « Separating Decision Diffie-Hellman from Diffie-Hellman in cryptographic groups ». *Cryptology ePrint Archive, Rapport 2001/003*, 2001. Disponible au http ://eprint.iacr.org/2001/003.

[KKSU00] N. Kanayama, T. Kobayashi, T. Saito, et S. Uchiyama. « Remarks on Elliptic Curve Discrete Logarithm Problems ». *IEICE Trans. Fundamentals*, vol. E83-A(01) :pp. 17–23, janvier 2000.

[Lan73] Serge Lang. « *Elliptic Functions* ». Addison-Wesley, 1973.

[LV01] A.K. Lenstra et E.R. Verheul. « Selecting Cryptographic Key Sizes ». *Journal of Cryptology*, vol. 14(4) :pp. 255–293, 2001.

[Lyn] Ben Lynn. « Elliptic Curves ». Disponible au : http ://rooster.stanford.edu/ ben/notes/elliptic/.

[Mag02] NETWORK Times Magazine. « M-commerce : the right technology for secure m-commerce payments ». (238), 2002. Disponible au : http ://networktimes.co.za/News.ASP ?pklNewsID=9122&pklIssueID=406.

[MEN93] Alfred J. MENEZES. « *Elliptic curve public key cryptosystems* ». Kluwer Academic Publishers, 1993.

[Mil86] Victor S. Miller. « Short Programs for Functions on Curves ». *Exploratory Computer Science IBM, Thomas J. Watson Research Center, Yorktown Heights, NY 10598*, May 1986.

[Mor93] Carlos J. Moreno. « *Algebraic Curves over Finite Fields* ». Cambridge University Press, 1993.

[MOV93] A. J. Menezes, T. Okamoto, et S. A. Vanstone. « Reducing Elliptic Curve Logarithms to Logarithms in Finite Field ». *IEEE Transactions on Information Theory*, vol. 39(05) :pp. 1639–1646, septembre 1993.

[MvOV97] A.J. Menezes, P.C. van Ooschot, et S. A. Vanstone. « *Handbook of Applied Cryptography* ». CRC Press, 1997.

[MW99] Ueli Maurer et Stefan Wolf. « The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms ». *SIAM Journal on Computing*, vol. 28(5) :pp. 1689–1721, 1999.

[Odl00] A. Odlyzko. « Discrete logarithms : The past and the future ». *Designs, Codes and Cryptography*, vol. 19(2-3) :pp. 129–145, 2000.

[ORT02] E. ORTIZ. « The wireless Messaging API ». *Sun Microsystems*, 2002. Disponible au : http ://developers.sun.com/techtopics/mobility/midp/articles/wma/index.html.

[PP03] K.G. Paterson et G. Price. « A comparison between traditional Public Key Infrastructures and Identity-Based Cryptography ». *Information Security Technical Report*, vol. 8(3) :pp. 57–72, 2003.

[Pro00] Java Community Process. « Mobile Information Device Profile (MIDP) ». (JSR-000037), 2000. Disponible au http ://jcp.org/aboutJava/communityprocess/final/jsr037/index.html.

[Pro03] Java Community Process. « Wireless Messaging API (WMA) for Java 2 Micro Edition, ver 1.1 ». (Release 120), 2003. Disponible au http ://jcp.org/aboutJava/communityprocess/final/jsr120/index2.html.

[RLB04] H. Ratshinanga, J. Lo, et J. Bishop. « A Security Mechanism for Secure SMS Communication ». *Computer Science Department, University of Pretoria, South Africa*, 2004. Disponible au : http ://www.cs.up.ac.za/cs/jbishop/Homepage/Pubs/Techreports/SecureSMS.pdf.

[SBK02] Rima Saliba, Gilbert Babin, et Peter Kropf. « SecAdvise : A Security Mechanism Advisor ». *in Conference on Distributed Communities on the Web (DCW 2002). Sydney, Australie. LNCS*, (2468) :pp. 35–40, Avril 2002.

[Sch84] A. Schamir. « Identity-Based Cryptosystems and Signature schemes ». *Advances in Cryptology - Crypto '84 , LNCS 196, Springer-Verlag*, pp. 47–53, 1984.

[Sil86] J.H. Silverman. « *The Arithmetic of Elliptic Curves* », Volume 106 *Graduate Texts in Mathematics.* Springer-Verlag, 1986.

[SL00] Kevin H. W. Shen et Daniel C. H. Lee. « WAP Mail Service and Short Message Service for Mobile CRM ». *International Symposium on Multimedia Software Engineering, IEEE*, pp. 201–207, 2000.

[Sol99] J. Solinas. « Generalized Mersenne numbers ». *Technical report CORR-39, Dept. of C&O, University of Waterloo*, 1999. Disponible au : http ://citeseer.ist.psu.edu/solinas99generalized.html.

[SU01] T. Saito et S. Uchiyama. « A Remark on the MOV Algorithm for Non-supersingular Elliptic Curves ». *IEICE Trans. Fundamentals*, vol. E84-A(05) :pp. 1266–1268, mai 2001.

[Tél] France Télécom. « Un pas supplémentaire dans le développement du M-Commerce ». Disponible au : http ://www.francetelecom.com/fr/entreprises/grandes_entreprises/ actualites/Transdev.html.

[Tre02] Paul Tremblett. « *Instant Wireless Java with J2ME* ». Osborne McGraw-Hill, 2002.

[Ver01] Eric R. Verheul. « Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems ». *Advances in Cryptology - Lecture Notes In Computer Science, Springer-Verlag*, vol. 2045 :pp. 195 – 210, 2001.

[Was03] Lawrence C. Washington. « *Elliptic Curves* ». CHAPMAN & HALL/CRC, 2003.

[Wen98] A. Weng. « Elliptic Curves ». Mars 1998. Disponible au http ://www.exp-math.uni-essen.de/ weng/project.ps.

[Whi01] James White. « An introduction to Java 2 micro edition (J2ME) ; Java in small ». *International Conference on Software Engineering, IEEE Computer Society*, pp. 724–725, 2001.

[Yac02] Yacov Yacobi. « A Note on the Bilinear Diffie-Hellman Assumption ». *Cryptology ePrint Archive, Rapport 2002/113*, 2002. Disponible au http ://eprint.iacr.org/2002/113.

[Yor92] E. V. York. « Elliptic Curves Over Finite Fields ». *George Mason University*, Mai 1992. Disponible au http ://www.mapleapps.com/categories/mathematics/algebra/code/elliptic/elliptic.pdf.

[Yua04] Michael Juntao Yuan. « *Enterprise J2ME : developing mobile Java applications* ». Prentice-Hall, 2004.

# Annexe A

# Exemple d'exécution et de fonctionnement

Nous présentons ci-après un exemple concret de fonctionnement du système de chiffrement FullIdent que nous avons implanté dans le chapitre 6. Pour chaque étape, nous donnons les principales instructions Java à exécuter suivi des résultats obtenus.

Rappelons que $q$ est un nombre premier de $k$ bits, $p$ est un nombre premier qui satisfait les conditions $p \equiv 3 \bmod 4$ et $q \mid (p^2 - 1)$ et $q \nmid p - 1$, $\mathbb{G}_1$ est le sous-groupe $E_{1,0}[q]$, $\mathbb{G}_2$ est le sous-groupe de l'extension du corps $\mathbb{F}_{p^2}$ d'ordre $q$.

**Setup :** Génération des paramètres du système *params* et la clé maître $s$

```
// instructions
int k = 98;
SParams params = new SParams(k);
params.compute();
ECPoint Ppub = params.getPointPpub();
ECPoint P = params.getPointP();
BigInteger q = params.getQ();
ECCurve curve = params.getCurve();
```

// retourne les parametres suivants

$q = 158456325028528675187087966209$

$p = 90532755950735310122167362130986449114344 3$

$P = 27312539291646048431112229157091155085918 3,$

$8388107826089972616174431903739627907796 28$

$Ppub = 24598076401165507957158067333755891520341 4,$

$6879059654401384818782459009166701770835 70$

// la clé maître, gardée secrètement par l'autorité PKG

$s = 121205200769855239645537426 82$

**Extract :** ECPoint $Qid = params.H_1("sms : // : +15149632045.6535"+"1204", curve, q)$ ;

// la clé privée sera valide seulement pendant le mois en cours

ECPoint Did = Qid.multiply(s) ;

//Résultats

$Qid = 66684632742923135420552071057819996131 4342,$

$5048844294437461026138223775007554579998 11$

$Did = 63354039764269011803480838854959694594526 0,$

$9577563839347954197625202512545686576469 8$

**Encrypt :** String Message = "Ceci est un message SMS"

byte cipherText[]=encrypt(params, Message, Qid) ;

la valeur aléatoire $r = 143403817963603912152772667775$

ECPoint Qid=params.H1("sms :// :+15149632045.6535"+ mois-courant(), curve , q) ;

la valeur du couplage de Tate $t_1 = t(P_{pub}, Q_{ID})^r$

=240494746192388050438221386298372801677266+
$i$70031954245991224233023811021626247917302

**Decrypt :** String ClearText = decrypt(params, cipherText, $d_{ID}$) ;
la valeur du couplage de Tate $t_2 = t(d_{ID}, U) =$
240494746192388050438221386298372801677266+
$i$70031954245991224233023811021626247917302
$= t_1$
ClearText = "Ceci est un message SMS"

**Vérification des propriétés mathématiques :**

$P, Ppub, Qid, Did \in \mathbb{G}_1$ sont tous des points de $q-$torsion : $[q]P = [q]Ppub = [q]Qid = [q]Did = \mathcal{O}$.

**La bilinéarité** du couplage est vérifiée : $t_1 = t_2$, voir ci-haut.

**q ième racine de l'unité :** La valeur du couplage $t_1 = t_2$ est supposée être la $q$ième racine de l'unité dans l'extension de corps $\mathbb{F}_{p^2}$. Effictivement, on a $t_1^q = 1 + i0$.