Université Cheikh Anta Diop de Dakar

École Doctorale Mathématiques et Informatique

N^o d'ordre : **155**

THÈSE DE DOCTORAT UNIQUE

Mention : *Mathématiques et Modélisation*
Spécialité : *Codage, Cryptographie, Algèbre et Applications*

Présentée et soutenue le 2 *novembre 2020 par*

Michel Seck

Pour obtenir le grade de

Docteur de l'Université Cheikh Anta Diop de Dakar

Titre : **Sur la construction de fonctions de hachage sur les courbes (hyper)elliptiques**

Devant le jury composé de

Président	:	Cheikh Thiécoumba GUEYE	Professeur Titulaire	Univ. Cheikh Anta Diop de Dakar
Rapporteurs	:	Luca De Feo	Maître de Conférences (HDR)	IBM Research GmbH, Zurich
		Youssouf Mahamadou DIAGANA	Professeur Titulaire	Univ. Nangui Abrogoua, Abidjan
		Mamadou SANGHARÉ	Professeur Titulaire	Univ. Cheikh Anta Diop de Dakar
Examinateurs	:	Ismaïla DIOUF	Maître de Conférences (CAMES)	Univ. Cheikh Anta Diop de Dakar
		Amadou Lamine FALL	Maître de Conférences (CAMES)	Univ. Cheikh Anta Diop de Dakar
Directeur de thèse	:	Djiby SOW	Professeur Titulaire	Univ. Cheikh Anta Diop de Dakar

Dédicaces

Je dédie ce modeste travail à

- ❀ mes parents Magate Seck et Sophie Kama qui m'ont accompagné, conseillé durant mes études depuis l'élémentaire ;
- ❀ mon oncle Docteur Ibou Seck pour son soutien considérable dans tous les cotés. Mention spéciale à ma tante Oumy Diao qui m'a accueilli à Dakar durant mon premier et second cycles universitaires ;
- ❀ mes frères, sœurs, cousins et cousines ;
- ❀ mes tantes et oncles ;
- ❀ mes chers professeurs ;
- ❀ mes étudiants (de TDSI, MPI et PCSM) de l'UCAD et de l'UVS.

Une dédicace spéciale à ma femme Awa Khady Sarr SECK. Un grand merci pour ta compréhension et de m'avoir soutenu dans tous les niveaux.

Remerciements

❀ J'adresse d'abord de chaleureux remerciements au Professeur Cheikh Thiécoumba Gueye de l'honneur qu'il m'a accordé d'avoir accepté de présider le jury de ma thèse. Merci Professeur de m'avoir initié en théorie des codes correcteurs d'erreurs.

❀ Je tiens à remercier mon directeur de thèse le Professeur Djiby Sow d'avoir accepté de travailler avec moi et pour sa générosité sans limite. Je lui suis également reconnaissant pour ses conseils dans tous les cotés (social, recherche, enseignement etc.), ses qualités pédagogiques et scientifiques, son humilité, sa franchise et sa sympathie. Je ne peux énumérer combien de fois, on est parti ensemble au restaurant pour déjeuner. Merci Professeur pour l'excellente formation que j'ai reçue auprès de vous et qui me sera certainement très utile pour ma future carrière.

❀ Je tiens à exprimer ma profonde gratitude à mes rapporteurs de thèse Prof. Lucas De Feo et Prof. Youssouf Mahamadou Diagana pour le temps considérable qu'ils ont consacré pour lire et corriger ce manuscrit. Une mention particulière à De Feo pour l'excellent cours de mathématiques pour les isogénies qu'il nous a donné lors de l'École Mathématique Africaine (EMA) de Thiès en 2017.

❀ Je remercie également les Professeurs Mamadou Sangharé, Ismaïla Diouf et Amadou Lamine Fall d'avoir accepté de faire partie des examinateurs de cette thèse. Un grand merci à Pr. Mamadou Sangharé d'avoir accepté d'animer des séminaires au laboratoire LACGAA pour nous, jeunes chercheurs. Merci à Prof Ismaïla de m'avoir initié en Python et à Prof Fall pour l'excellent cours d'algèbre en première année MPI qu'il nous a donné.

❀ Mes remerciements à tous les membres du Laboratoire d'Algèbre de Cryptographie, de Géométrie Algébrique et Applications (LACGAA), particulièrement, Pr. Oumar Diankha, Pr. Sidy Demba Touré, Pr. Maouia Ben Faraj, Pr. Mamadou Barry, Pr. Leila Mesmoudi, Pr. Abdoulaye Mbaye, Dr. Amadou Tall, Dr. Ousmane Ndiaye, Dr. Anta Niane, Dr. Jean Bello Klamti, Dr. Demba Sow, Dr. Alassane Traoré, Dr. Chérif Bachire Dème, Dr. Ahmed Youssef Khlil, M. Adjeoua etc. Merci également à tous les autres professeurs du Département de Mathématiques et Informatique, en particulier Prof. Masseye Gaye.

❀ Je remercie également tous les autres membres du groupe de recherche de Mathématiques Discrètes et Cybersécurté (MDC) dont je fais partie : Soda Diop, Stone Guy Wamba, Mme Ba, Sidoine, Pape Modou Ndiaye, Bernard Ousmane Sané, Clément.

❀ Merci à tous les doctorants de l'Ecole Doctorale de Mathématiques et Informatique (EDMI) qui m'ont choisi comme représentant des doctorants au conseil scientifique et pédagogique de l'EDMI. Je les remercie également pour les moments d'échanges inoubliables lors des séminaires des doctorants et jeunes chercheurs. J'en profite pour remercier notre Directeur de l'EDMI, le Professeur Hamidou Dathe et notre Curateur de Thèse, le Professeur Djaraf SECK.

❀ Merci également au personnel de la scolarité de la FST, du secrétariat du département de Mathématiques et Informatique, de la licence et du master en Transmission de Données et Sécurité de l'Information (TDSI). Je veux nommer Mme Samb Oulimata Lydia, Mme Diouf, Mme Ndeye Samb, Mr Boubacar Sow, Mme Mbaye, Mme Ba, Mme Gnima, Mr Massaly, Mme Toukara.

❀ Merci à mes collègues tuteurs de AIMS : Dr. Cheikh Birahime Ndao, Dr. Ignace Aristide Milinde, Dr. Oumy Niass, Dr. Saliou Diouf, Dr. Mouhamad M. Allaya, Dr. Amy Sadio, Dr. Abou Sène, Dr. Jean Claude Outaziroubanda, M. Mbaye faye.

❀ Mes sincères remerciements au Pôle de Recherche en Mathématiques et leurs Applications à la Sécurité de l'Information (PRMASI) dont je suis membre, pour leur soutien matériel et financier (ordinateur, bourse) et de m'avoir permis de participer à plusieurs Écoles Mathématiques Africaines (EMA). Un grand merci à tous les membres du pôle : Dr. Tony Ezomé, Dr. Abdoul Aziz Ciss, Dr. Emmanuel Fouotsa, Dr. Hortense Boudjou, Dr. Nafissatou Diarra, Dr. Thierry Mefenza Nountu, Dr. Aminatou Pecha, Dr. Herve Kalachi Talé, Dr. Aboussaghid Alkabouss Sakha, Mouhamadou Sall, Abdoulaye Maiga.

❀ Je remercie vivement le Centre d'Excellence Africain en Mathématiques, Informatique et Technologies de l'Information et de la Communication (CEA-MITIC) de m'avoir accordé une prise en charge totale pour ma participation à la conférence internationale de cryptologie en Afrique (AFRICACRYPT-2018) à Marrakech.

❀ Merci enfin, à tous ceux dont je n'ai pas cité les noms et qui ont contribué, de près ou de loin, à mon parcours.

Table des matières

Chapitre 1

Introduction et Synthèse de nos travaux

Contents

La sécurité de l'information est un enjeu à la fois pour le citoyen, l'entreprise et l'état. L'information est précieuse, il faut donc la protéger. C'est là où intervient la cryptographie. Beaucoup de protocoles et d'algorithmes cryptographiques ont été mis en place pour régler la plupart des questions sécuritaires des systèmes d'information.

1.1 Cryptographie

La cryptologie [MVO96] est une science des mathématiques qui étudie la sécurité de l'information et des communications ; elle se subdivise en deux branches :

- **la cryptographie** : une science qui s'intéresse à la conception de schémas et protocoles qui permettent de garantir la sécurité de l'information et des communications en présence d'un potentiel adversaire. Le mot cryptographie vient des mots grecs *kryptós* qui signifie «caché, secret» et *gráfein* qui signifie «écrire» ;
- **la cryptanalyse** : une science qui étudie les attaques contre les schémas cryptographiques. Une hypothèse fondamentale en cryptanalyse a été formulée par A. Kerkhoff au XIXe siècle (en 1884). Il est généralement appelé le principe de Kerkhoff. Il stipule que l'adversaire connaît tous les détails du schéma cryptographique, y compris les algorithmes et leurs implémentations. Selon ce principe, la sécurité d'un schéma de chiffrement doit être entièrement basée sur la clef secrète.

Le cryptographe donc propose des algorithmes à exécuter par les différentes parties pour satisfaire des propriétés de sécurité, et le cryptanalyste y cherche des faiblesses. Parmi les propriétés de sécurité prises en charge par la cryptographie, on peut citer :

- **la confidentialité :** c'est une propriété qui assure qu'une information n'est pas révélée à des entités non autorisées.
- **l'intégrité des données :** c'est un service qui assure l'authenticité de l'information. Ce service permet donc de garantir qu'une information n'a pas été modifiée.
- **l'authentification :** c'est un service qui permet de prouver l'identité de l'un des partenaires d'une communication.
- **la non-répudiation :** c'est un service qui permet d'empêcher une entité de nier ses engagements ou ses actions précédentes. Par exemple, si une entité $\mathbb{A}$ envoie à une autre entité $\mathbb{B}$ un message $\mathbf{M}$, alors $\mathbb{A}$ ne doit pas, dans le futur, nier que le message $\mathbf{M}$ provient d'elle.

Le but traditionnel de la cryptographie était d'élaborer des méthodes permettant de transmettre des données de manière confidentielle. Le problème qu'on cherche à résoudre est le suivant : deux personnes A et B veulent échanger des informations confidentielles sous la présence potentielle d'un adversaire E [Tho00]. Des réponses ont été proposées depuis l'antiquité en faisant recourt aux *codes secrets* qui ont permis de décider du sort des hommes et nations, par exemple, l'Enigma a joué un rôle central durant la seconde guerre mondiale.

En ce qui concerne la cryptographie moderne, le but n'est pas seulement d'assurer la confidentialité des communications, mais aussi l'intégrité des données, l'authentification

des divers acteurs, la non-répudiation d'un contrat numérique (la signature numérique), la certification, le contrôle d'accès, la preuve de connaissance etc.

Par abus de langage, les cryptographes désignent généralement les deux parties d'une communication par **Bob** et **Alice** et l'adversaire (l'attaquant) par **Ève** même si dans la vie réelle ils représentent généralement des ordinateurs, des serveurs, des téléphones, des cartes bancaires etc.

1.1.1 Échange de clefs Diffie-Hellman

Diffie et Hellman [DH06], en 1976, ont énoncé un principe permettant à deux partenaires d'une communication Alice et Bob de partager une information secrète uniquement à partir de données publiques. C'est le début de la cryptographie à clefs publiques. Le principe fonctionne comme suit :

- Alice et Bob s'accordent sur un groupe (ou sous groupe) cyclique $\mathbf{G}$ (noté multiplicativement) d'ordre premier $\mathbf{p}$ et un élément générateur $\mathbf{g} \in \mathbf{G}$;
- Alice choisit un entier aléatoire $\mathbf{a} < p$, calcule ensuite $\mathbf{k_A} = \mathbf{g^a}$;
- Bob choisit un entier aléatoire $\mathbf{b} < p$, calcule ensuite $\mathbf{k_B} = \mathbf{g^b}$;
- Alice (resp Bob) récupère $\mathbf{k_B}$ (resp $\mathbf{k_A}$) puis calcule $\mathbf{k} = (\mathbf{k_B})^\mathbf{a}$ (resp $\mathbf{k'} = (\mathbf{k_A})^\mathbf{b}$).
- Ainsi $\mathbf{g^{ab}} = (\mathbf{k_A})^\mathbf{b} = (\mathbf{k_B})^\mathbf{a}$ est la clef commune de Alice et Bob.

On peut se demander, si un attaquant intercepte les clefs $\mathbf{k_A}$ et $\mathbf{k_B}$ peut retrouver le secret commun $\mathbf{g^{ab}}$. Le problème d'extraction de l'information $\mathbf{x}$ à partir des informations $\mathbf{g}$ et $\mathbf{g^x}$ dans $\mathbf{G}$ est appelé problème du logarithme discret (PLD) dans G. Il existe deux autres problèmes liés au logarithme discret : les problèmes **calculatoire** et **décisionnel** de Diffie-Hellman.

- **Problème calculatoire de Diffie-Hellman**(CDH) : c'est de calculer l'information $\mathbf{g^{ab}}$ connaissant les informations $\mathbf{g}$, $\mathbf{g^a}$ et $\mathbf{g^b}$.
- **Problème décisionnel de Diffie-Hellman**(DDH) : c'est de distinguer les deux distributions $(\mathbf{g^a}, \mathbf{g^b}, \mathbf{g^{ab}})$ et $(\mathbf{g^a}, \mathbf{g^b}, \mathbf{g^c})$ où c est généré aléatoirement $(c < p)$.

Le protocole d'échange de clefs de Diffie-Hellman, tel qu'il est présenté en 1976 par Whitfield Diffie et Martin Hellman, est vulnérable. En effet, un attaquant Ève peut se mettre entre Alice et Bob et modifier toutes les informations échangées : c'est l'attaque de l'homme du milieu.

Attaque de l'homme du milieu :

Si Ève intercepte la valeur $\mathbf{g^a}$ envoyée par Alice à Bob, il peut la transformer en une autre valeur $\mathbf{g^{a'}}$ et se faire passer pour Alice. De même, Ève peut intercepter la valeur $\mathbf{g^b}$ envoyée par Bob à Alice et la changer en $\mathbf{g^{b'}}$ et se faire passer cette fois-ci pour Bob. Ainsi Ève aura un secret qu'il partage avec Alice $\mathbf{g^{b'a}}$ et un autre qu'il partage avec Bob $\mathbf{g^{a'b}}$. Ève pourra donc déchiffrer n'importe quel message envoyé par Alice ou Bob.

Mais heureusement, il y'a une solution à cette attaque. Elle consiste à signer les messages échangés à l'aide d'une paire de clefs asymétriques certifiées par une autorité de confiance. Bob peut ainsi être assuré que la clef qu'il reçoit provient effectivement de

Alice, et réciproquement pour Alice.

Pour la sécurité d'un protocole cryptographique basé sur le problème du logarithme discret, il faut absolument choisir un groupe G où la résolution de l'équation $a = g^x$ est très difficile. D'autres groupes ont été proposés parmi lesquels on peut citer le groupe attaché à une courbe elliptique ou hyperelliptique. La version du protocole d'échange de clefs Diffie-Helmann sur les courbes elliptiques (Elliptic Curve Diffie-Helmann) est présentée à l'annexe B à la section B.3.6. Il est à noter que le PLD ne résiste pas à l'ordinateur quantique [KM16].

1.1.2 Fonctions de hachage

Beaucoup de schémas cryptographiques utilisent des fonctions de hachage, on peut citer, entre autres, le schéma de chiffrement basé sur l'identité de Boneh et Franklin. Dans leur schéma, Boneh et Franklin [BF01] utilisent une fonction de hachage publique pour générer la clef publique.

Une fonction de hachage h est une fonction qui fait correspondre à une entrée de taille quelconque (une chaine de bits) une sortie de taille fixe aléatoire. i.e $h : \{0,1\}^\star \to \{0,1\}^n$: $m \longmapsto h(m)$ où $\{0,1\}^\star$ désigne une chaine de bits de longueur quelconque et $\{0,1\}^n$ une chaine de bits de longueur n. La sortie $h(m)$ est appelée empreinte digitale ou haché. Pour une fonction de hachage cryptographique, ça ne doit pas être possible de trouver certaines relations entre l'entrée et la sortie, ou de trouver des sorties avec des relations particulières autrement que par recherche exhaustive. Une fonction de hachage (cryptographique) doit donc avoir les propriétés suivantes :

- L'empreinte d'un message doit être facile à calculer,
- Étant donnée une chaine de bits $y \in \{0,1\}^n$, il est difficile de trouver $m \in \{0,1\}^\star$ tel que $h(m) = y$, c'est-à-dire pour n'importe quel haché y, un attaquant ne doit pas être en mesure de trouver un message m tel que $h(m) = y$ en un temps polynomial ou sous-exponentiel,
- Résistante aux collisions faibles : Connaissant un message m, il est difficile de trouver un autre message m' tel que $h(m) = h(m')$,
- Résistante aux collisions fortes : Il est très difficile de trouver deux message m et m' ayant la même empreinte.

Ainsi, pour une bonne fonction de hachage (cryptographique), il doit être impossible en pratique de trouver des collisions ou des antécédents. Les fonctions de hachage ont de nombreuses utilisations en informatique, basées sur l'idée que l'empreinte peut servir à identifier le message sous une forme plus compacte. Parmi ces applications, on peut citer :

- ▶ **protection des mots de passe** : on ne stocke pas directement les mots de passe mais leurs hachés ;
- ▶ **confirmation sans divulgation de connaissance** : si on veut prouver à quelqu'un qu'on connait un secret sans le révéler dans l'immédiat, on peut publier le haché de ce secret. Une fois le secret révélé, il est facile de vérifier ses dires ;

- **génération ou dérivation de clefs** : les fonctions de hachage permettent de casser la structure algébrique et statistique de l'élément haché, donc elles permettent de générer des chaines aléatoires avec une preuve dans le modèle de l'oracle aléatoire ;
- **signature numérique** : pour des raisons d'efficacité et de sécurité, on ne signe jamais directement un message M mais toujours son haché ;
- **intégrité de communications** : une fonction de hachage avec une clef permet de fabriquer un MAC (Message Authentication Code) et donc de garantir qu'un message n'a pas été modifié en cours de transmission.
- **blockchain** : une blockchain comporte une séquence de plusieurs blocs et chaque bloc contient le haché d'informations du bloc précédent ;
- **preuves de sécurité dans le modèle de l'oracle aléatoire** : les techniques [Den03, FO13, HHK17] qui permettent de transformer un schéma de chiffrement (voir Section 1.1.4) ayant la sécurité CPA en un mécanisme d'encapsulation de clef (voir Section 1.1.5) ayant le niveau de sécurité CCA2 dans le modèle de l'oracle aléatoire utilisent des fonctions de hachage.

Exemples de fonctions de hachages publiques : MD5, SHA3 et SHAKE.

1.1.3 Systèmes symétriques et systèmes asymétriques

En cryptographie, on a deux types de systèmes de chiffrement : les systèmes symétriques et les systèmes asymétriques.

Systèmes symétriques

Le système symétrique, également dit à clef secrète, est la plus ancienne forme de chiffrement. Dans ce système, les clefs de déchiffrement et de chiffrement sont soit identiques, soit chacune des clefs est facilement retrouvable à partir de l'autre. Pour faire parvenir un message de façon sûre, il faut le chiffrer à l'aide d'une clef connue uniquement de l'expéditeur et du destinataire, puis faire parvenir au destinataire prévu le message. Beaucoup d'algorithmes sont basés sur ce système, parmi lesquels, on peut citer : le chiffrement de César et de Vigenère et de façon générale les schémas de chiffrement classiques, DES (Digital Encryption Standard), Triple DES ou 3DES, AES (Advanced Encryption Standard), IDEA (International Data Encryption Algorithm) etc.

Dans le système de chiffrement symétrique, le principal avantage qu'on obtient est la rapidité. Par contre on est confronté à quatre problèmes majeurs :

- **le partage des clefs secrètes** : on résout ce problème, en utilisant la valise diplomatique ou le protocole d'échange de clefs de Diffie-Hellman ;
- **la gestion des clefs secrètes** : on résout ce problème, en évitant d'utiliser une clef plus d'une fois et en utilisant des coffres spécialisés si on est obligé de garder les clefs assez longtemps comme dans le cas de l'archivage des données secrètes ;
- **la distribution des clefs** : supposons que $\mathbf{n}$ parties veulent communiquer entre elles, alors il va falloir créer $\frac{\mathbf{n(n-1)}}{\mathbf{2}}$ clefs et chaque participant devant stocker

$n-1$ clefs. Ce qui est difficile à gérer lorsque n devient très grand. Un système asymétrique règle une bonne partie de ce problème (voir ci-dessous) ;

- **la non-répudiation** : Supposons que Alice et Bob ont une clef commune K. Donc Alice et Bob ont les mêmes capacités car possédant la même clef. Bob peut envoyer un message chiffré avec la clef K et changer d'avis plus tard et nier l'avoir émis ; il peut prétendre que le message provient d'Alice. On résout ce problème en utilisant un système asymétrique à travers les signatures numériques.

Systèmes asymétriques

Le système asymétrique encore appelé système à clef publique repose sur des problèmes mathématiques réputés difficiles à savoir :

— le problème de la factorisation des grands entiers,
— le problème du logarithme discret dans des groupes,
— les problèmes liés à la résiduosité quadratique,
— le problème du vecteur le plus court ou du vecteur le plus proche dans un réseau arithmétique,
— le problème de décodage d'un code correcteur aléatoire,
— le problème de calcul d'isogénie entre courbes elliptiques.

Dans ce système, on utilise une paire de clefs (p_k, s_k) dont l'une est difficile à obtenir à partir de l'autre contrairement au système précédent. La clef qui est difficile à obtenir à partir de l'autre est gardée secrète (ça sera la clef privée s_k) et n'est connue que par une seule personne, tandis que la seconde est mise à la disposition de tout le monde (c'est la clef publique p_k).

Pour faire parvenir à une entité un message m (assez court) de façon sûre, on utilise la clef publique p_k du destinataire pour chiffrer et lui faire parvenir ce message chiffré (c) et à la réception, ce dernier utilise à son tour sa clef privée s_k pour déchiffrer c et obtenir m. Parmi les algorithmes de chiffrement à clef publique, on peut citer :

- **RSA** (Rivest, Shamir et Adleman) en 1977 [RSA78] : c'est un des algorithmes les plus utilisés au monde durant les deux dernières décennies. Ce système de chiffrement est basé sur la difficulté de factoriser les grands entiers ;
- **Elgamal** : Il est fondé sur le problème du logarithme discret et créé par Tahel Elgamal en 1978 [Gam85] ;
- **Ntru** (introduit par Jeffrey Hoffstein, Joseph H. Silverman et Jill Pipher [HPS98] en 1996) : il est lié au problème du vecteur court et du vecteur le plus proche dans un réseau euclidien.

Dans ce système à clef publique, on rencontre quelques problèmes à savoir :

1. la lenteur des algorithmes de chiffrement, c'est la raison pour laquelle, ces systèmes sont utilisés en général seulement pour le transfert de clefs secrètes et pour la signature numérique ;
2. l'authenticité des clefs publiques : on résout ce problème en faisant appel à un tiers de confiance appelé autorité de certification.

1.1.4 Schémas de chiffrement

- **fonction à sens unique** [MVO96] : une fonction d'un ensemble A vers un ensemble B est appelée fonction à sens unique si $f(a)$ est facile à calculer pour tout $a \in A$, mais pour un b choisi aléatoirement dans l'image de f, trouver un $a \in A$ tel que $b = f(a)$ est calculatoirement impossible (en temps polynomial). En d'autres mots, on peut facilement calculer $f(x)$ pour tout $x \in A$, mais il est (presque) impossible de calculer $f^{-1}(y)$ pour un $y \in B$.
- **fonction à sens unique avec trappe** : une fonction $f : A \to B$ est dite à sens unique avec trappe si
  - f est une fonction à sens unique,
  - il existe une information supplémentaire, appelée trappe telle que pour un b donné dans $\mathrm{Im}(f)$, il est facile de trouver $a \in A$ tel que $f(a) = b$, mais sans la trappe, il est impossible de trouver un antécédent de b (en temps polynomial).

Soient $\mathcal{M}$ l'espace des messages à chiffrer, $\mathcal{C}$ l'espace des chiffrés et $\mathcal{K}$ l'espace des clefs. Un schéma de chiffrement est un triplet $(\mathcal{K}\mathrm{eygen}, \mathrm{Chif}_k, \mathrm{Dec}_k)$ où

- $\mathcal{K}$eygen est un algorithme de génération de clefs qui génère des clefs secrètes $s_k \in \mathcal{K}$ dans le cas d'un système symétrique et de paires de clefs $(p_k, s_k) \in \mathcal{K} \times \mathcal{K}$ où p_k est une clef publique et s_k est une clef privée dans le cas d'un système asymétrique ;
- Chif_k est une fonction à sens unique avec trappe définie de $\mathcal{M} \times \mathcal{K}$ dans $\mathcal{C}$ appelée algorithme de chiffrement.
- Dec_k est une fonction définie de $\mathcal{C} \times \mathcal{K}$ dans $\mathcal{M}$ appelée algorithme de déchiffrement telle que, pour presque tout message $m \in \mathcal{M}$, $\mathrm{Dec}_k\left(\mathrm{Chif}_k(m, k_c), k_d\right) = m$ si k_c est une clef de chiffrement et k_d une clef de déchiffrement associée à k_d. Dans le cas d'un système symétrique, généralement, $k_c = k_d$ et dans le cas d'un système asymétrique, $k_c = p_k$ est la clef publique du destinataire du message m et $k_d = s_k$ est la clef privée du destinataire.

Un schéma de chiffrement à clef publique (PKE) est dit $\delta-$correct si (la probabilité)

$$Pr\left[\mathrm{Dec}_k\left(\mathrm{Chif}_k(m, p_k), s_k\right) \neq m\right] \leqslant \delta$$

pour toute paire de clefs $(p_k, s_k) \in \mathcal{K} \times \mathcal{K}$ et pour tout message $m \in \mathcal{M}$. Si $\delta = 0$ i.e $Pr\left[\mathrm{Dec}_k\left(\mathrm{Chif}_k(m, p_k), s_k\right) = m\right] = 1$, on dira que le PKE est correct. Il faut noter qu'un PKE correct signifie qu'il ne peut y avoir d'échecs de déchiffrement.

1.1.5 Mécanisme d'encapsulation de clef (KEM)

Soient $\mathcal{C}$ l'espace des chiffrés et $\mathcal{K}$ l'espace des clefs. Un mécanisme d'encapsulation de clef (Key Encapsulation Mechanism (KEM) en anglais) [BBF+18, BDK+17, Den03] est un triplet d'algorithmes (**Keygen**, **Encaps**, **Decaps**) où

- **Keygen** est un algorithme (probabiliste) de génération de paires de clefs $(p_k, s_k) \in \mathcal{K} \times \mathcal{K}$ où p_k est une clef publique et s_k une clef privée.
- **Encaps** est un algorithme (probabiliste) d'encapsulation qui prend en entrée une clef publique p_k et retourne un chiffré $c \in \mathcal{C}$ et une clef $k \in \mathcal{K}$.

- **Decaps** est un algorithme (déterministe) de décapsulation qui prend en entrée une clef privée s_k et un chiffré c et retourne une clef $k \in \mathcal{K}$ or $\perp$ (qui signifie échec).

Un KEM est dit $\varepsilon-$correct si pour toute paire de clefs $(p_k, s_k) = \textbf{Keygen}()$ et pour tout couple $(c, k) = \textbf{Encaps}(p_k)$ alors la probabilité $Pr\left[\textbf{Decaps}(s_k, c) \neq k\right] \leqslant \varepsilon$. On dit qu'il est correct lorsque $\varepsilon = 0$.

PKE $\overset{vers}{\Longrightarrow}$ **KEM** : Il existe des mécanismes de transformation d'un système de chiffrement asymétrique en un mécanisme d'encapsulation de clef. Certaines transformations requièrent que le schéma de chiffrement (PKE) soit correct, c'est-à-dire, qu'il n y ait pas d'échecs de déchiffrement ; on peut citer, entre autres, les transformations d'Alexander W. Dent [Den03] et ceux de Fujisaki-Okamoto (FO) [FO13]. En 2017, Hofheinz, Hövelmanns and Kiltz [HHK17] révisent les transformations de FO et en proposent de nouvelles qui ne nécessitent pas que le schéma soit correct mais seulement $\delta-$correct. A l'appel de NIST [Nist] pour des schémas post-quantiques, 39 KEMs ont été proposés et la plupart d'entre eux proviennent de telles transformations.

1.1.6 Schémas de signature

Soient $\mathcal{M}$ l'espace des messages, $\mathcal{K}$ l'espace des clefs et $\mathcal{S}$ un ensemble d'éléments (habituellement de chaines de bits) de taille fixe appelé espace des signatures. On pose $\mathcal{M}' = \{h(m), m \in \mathcal{M}\}$ où h est une fonction de hachage. Un schéma de signature numérique est un triplet $(\mathcal{K}\text{eygen}, \text{Sig}_{s_k}, \text{Ver}_{p_k})$ où

- $\mathcal{K}$eygen est un algorithme de génération de paires de clefs $(p_k, s_k) \in \mathcal{K} \times \mathcal{K}$ où p_k est une clef publique et s_k une clef privée,
- Sig_{s_k} est une fonction définie de $\mathcal{M}$ dans $\mathcal{S}$ (pour une clef privée s_k) appelée algorithme de génération de signatures,
- Ver_{p_k} est une fonction définie de $\mathcal{M}' \times \mathcal{S}$ dans $\mathbb{F}_2 = \{0, 1\}$ appelée algorithme de vérification telle que $\text{Ver}_{p_k}(h(m), c) = 1$ si $\text{Sig}_{s_k}(h(m)) = c$ et $\text{Ver}_{p_k}(h(m), c) = 0$ dans le cas contraire.

Un schéma de signature avec *récupération de message* est un schéma de signature pour lequel le message envoyé n'est pas requis en entrée de l'algorithme de validation (ou de vérification). Dans ce cas, le message d'origine est récupéré à partir de la signature elle-même (par exemple le schéma de signature RSA [RSA78]). Un schéma de signature *avec appendice* est un schéma de signature où le message est requis comme entrée pour l'algorithme de vérification (exemples : les schémas de signature ElGamal [Gam85], DSA et Schnorr).

Alice disposant d'une paire de clefs (p_k, s_k) publique et privée (produite par l'algorithme de génération de clefs $\mathcal{K}$eygen), si elle veut signer un message m, elle utilise sa clef privée s_k et fait appel à l'algorithme de génération de signatures pour obtenir une signature $\text{Sig}_{s_k}(h(m)) = c$ et toute entité disposant de la signature (m, c) et de la clef publique p_k d'Alice doit pouvoir vérifier que la signature (m, c) est valide ou non en utilisant l'algorithme de vérification $\text{Ver}_{p_k}(h(m), c)$.

Outre d'être **publiquement vérifiable**, un bon schéma de signature numérique doit avoir les propriétés suivantes.

- ✔ **Inforgeable** : si Alice signe un message m avec $\mathrm{Sig}_{s_k}(h(m))$, elle doit être calculatoirement infaisable pour un adversaire Eve ne disposant pas la clef privée d'Alice p_k de générer une paire $(m, \mathrm{Sig}_{s_k}(h(m)))$ valide. Donc Alice ne doit pas pouvoir nier une signature après l'avoir transmise (**non-répudiation**) car il doit être impossible de forger une signature valide.
- ✔ **Authentique** : si Bob reçoit $\mathrm{Sig}_{s_k}(h(m)) = c$ d'Alice, alors Bob doit pouvoir vérifier que la signature c provient bel et bien d'Alice en utilisant $\mathrm{Ver}_{p_k}(h(m), c)$.
- ✔ **Non-réutilisable** : si Alice signe un message m avec $\mathrm{Sig}_{s_k}(h(m)) = c$, alors c ne peut plus être une signature valide pour un autre message $m' \neq m$.

Un schéma de signature numérique est donc un service cryptographique analogue aux signatures manuscrites mais qui apporte plus de sécurité.

1.1.7 Sécurité prouvée

Il y'a quelques décennies, Il était de coutume, en cryptographie, d'utiliser un schéma qui n'a pas été prouvé sûr juste parce qu'aucune attaque efficace n'est connue. Comme l'absence d'attaque ne peut justifier la sureté d'un schéma cryptographique, les cryptographes ont mis en place un formalisme mathématique rigoureux de la sécurité. Il est important de noter qu'un schéma cryptographique prouvé mathématiquement sûr ne veut pas dire qu'il n'y aurait pas d'attaques sur un tel schéma, mais ça permet de relier la cassabilité de ce schéma à la résolution d'un problème mathématique réputé difficile.

Notions de sécurité

Une notion de sécurité est un couple composé d'un objectif de sécurité et d'un modèle d'attaquant [Sow16].

— L'objectif de sécurité spécifie ce que l'on souhaite concrètement protéger dans un cryptosystème (par exemple pour la cryptographie à clef publique, on peut citer les objectifs suivants : Incassabilité, Fonction à sens unique, Indistinguabilité et Non-Malléabilité).

— Le modèle de l'attaquant spécifie les moyens et la puissance de calcul supposés être à la disposition de l'attaquant pour essayer de faire le calcul que nécessite son attaque.

Modèles de preuves de sécurité

Oracle : Un oracle est un algorithme $\mathcal{A}$ auquel on peut soumettre une entrée x et obtenir la sortie $\mathcal{A}(x)$ en boite noire (autrement dit, on ne maitrise pas du tout comment $\mathcal{A}$ calcule $\mathcal{A}(x)$).

Fonction aléatoire : Une fonction $f : \{0,1\}^n \to \{0,1\}^*$ est dite aléatoire si pour tout $x \in \{0,1\}^n$, chaque bit de la suite $f(x)$ est aléatoire (c'est-à-dire que les bits 0 et 1 sont équiprobables, quelque soit la position de chacun). L'ensemble $\{0,1\}^n$ désigne l'ensemble des suites finies de bits et $\{0,1\}^*$ l'ensemble des suites infinies de bits.

Oracle aléatoire : Un oracle aléatoire est un oracle $\mathcal{O}$ qui est en même temps une fonction aléatoire.

☛ **Preuve dans le modèle de l'oracle aléatoire** : si on fait appel à des oracles aléatoires, dans une preuve de sécurité, on dit que la preuve se fait dans le modèle de l'oracle aléatoire et dans ce cas ces oracles aléatoires correspondent à des fonctions de hachage dans le schéma.

☛ **Preuve dans le modèle standard** : si on n'utilise pas d'oracles aléatoires, on dit qu'on est dans le modèle standard.

La plupart des preuves de sécurité des schémas cryptographiques sont dans le modèle de l'oracle aléatoire. Cependant il faut noter qu'une preuve dans le modèle oracle aléatoire ne garantit pas que le système est sûr dans la vie réelle, mais néanmoins ce procédé de preuve dans le modèle de l'oracle aléatoire est unanimement admis.

Sécurité d'un système de chiffrement à clef publique

Quand on construit un schéma de chiffrement à clef publique, les objectifs visés sont :

❀ **l'Incassabilité** : sachant qu'il y'a une paire de clefs (p_k, s_k) publique et privée et que la clef publique p_k est supposée connue de tous et la clef privée d'une seule entité, il doit être impossible de déduire la clef privée (en temps polynomial) à partir de la clef publique et d'un chiffré, sinon le schéma est totalement cassé.

❀ **fonction à sen unique** : sachant que l'objectif de chiffrer un message est que seul le destinataire est destiné à en prendre connaissance, la fonction de chiffrement doit être à sens unique, donc on ne doit pas pouvoir déduire un clair de son chiffré (en temps raisonnable).

❀ **l'Indistinguabilité** : connaissant le chiffré c d'un clair m, on ne doit pas pouvoir déduire aucune information sur m ne serait-ce qu'un seul bit d'information de m (d'une meilleure façon qu'un choix au hasard). En pratique si l'attaquant donne deux messages m_1 et m_2 et qu'on en chiffre un au hasard en c, alors l'attaquant ne peut pas savoir lequel est chiffré en c.

❀ **la Non-Malléabilité** : on ne doit pas pouvoir transformer un chiffré c_1 d'un clair m_1 en un chiffré c_2 d'un clair m_2 de sorte que m_1 et m_2 soient reliés avec une probabilité meilleur que la probabilité uniforme (absence de corrélations).

Parmi les attaques que peut subir un schéma de chiffrement à clef publique, il y a :

CPA (Attaque à textes clairs choisis : Chosen-Plaintext Attack) : l'attaquant a accès à un oracle de chiffrement $\mathcal{O}_{chif}$, donc il peut obtenir les chiffrés de son choix. Ce qui est trivial en cryptographie à clef publique

CCA1 (Attaque **non** adaptative à chiffrés choisis : (Non-Adaptive) Chosen-CipherText Attack) : En plus d'avoir accès à un oracle de chiffrement $\mathcal{O}_{chif}$, l'attaquant a accès à un oracle de déchiffrement $\mathcal{O}_{dechif}$ avant de recevoir le challenge, c'est à dire, le chiffré à attaquer.

CCA2 (Attaque adaptative à chiffrés choisis : (Adaptive) Chosen-CipherText Attack) : En plus d'avoir accès à un oracle de chiffrement $\mathcal{O}_{chif}$, l'attaquant peut faire appel à un oracle de déchiffrement avant ($\mathcal{O}_{dechif1}$) et après ($\mathcal{O}_{dechif2}$) avoir

reçu le challenge. **Mais le challenge n'est pas soumis à l'oracle**. Cette attaque est plus forte que les précédentes car après avoir pris connaissance du challenge il peut adapter les chiffrés qu'il souhaite faire déchiffrer.

Sécurité d'un système de signature

Objectifs de sécurité : quand on construit un schéma de signature numérique, les principaux objectifs visés sont l'incassabilité et l'inforgeabilité.

Moyens de l'attaquant : Étant donné que la clef publique est supposée connue de tous, on suppose donc que l'attaquant peut vérifier si une signature est valide ou non. Les moyens de l'attaquant dépendent donc du niveau d'accès aux messages signés. On a les scénarios suivants [Kat10] :

KMA (attaque à messages connus (Known-message attack)) : l'attaquant a un accès limité à certains messages signés. Il peut obtenir les signatures de certains messages avant de voir la clef publique.

CMA (attaque (adaptative) à messages choisis : (Adaptive) Chosen-message attack) : c'est le mode d'attaque le plus élevé. L'attaquant a accès à un oracle de génération de signatures $\mathcal{O}_{sign}$ non seulement avant de voir la clef publique mais il peut choisir des messages à signer basés sur les messages précédemment signés.

1.1.8 Cryptographie post-quantique

La sécurité des communications sur internet d'aujourd'hui est généralement assurée par des primitives cryptographiques telles que RSA et ECC qui sont cassées théoriquement par la machine quantique. Ces primitives et généralement tous les schémas cryptographiques dont la sécurité est basée sur le problème de factorisation des grands entiers et sur le logarithme discret pourraient être remplacés dans quelques années à cause de l'algorithme quantique de Shor (1994) [Sho94, Sho97]. Certains schémas de chiffrement symétriques, comme AES, seront affectés ; les tailles de clefs pourraient être doublées, triplées ou même plus à cause de l'algorithme quantique de Grover [Gro96].

Il y'a quelques années, à cause des algorithmes quantiques, on pensait que la cryptographie basée sur les courbes elliptiques et hyperelliptiques serait morte. Il est admis aujourd'hui que la cryptographie basée sur le calcul d'isogénies entre courbes elliptiques (et généralement entre Jacobiens de courbes elliptiques ou hyperelliptiques) résiste à la machine quantique. Pour les autres domaines post-quantiques [KM16], outre la cryptographie basée sur le calcul d'isogénies, on peut citer :

- la cryptographie symétrique
- la cryptographie basée sur les réseaux arithmétiques (euclidiens),
- la cryptographie basée sur les codes correcteurs d'erreurs,
- la cryptographie basée sur les polynômes multivariés,
- la cryptographie basée sur le calcul quantique,
- les fonctions de hachage.

La communauté cryptographique et les organismes de sécurité se préparent déjà à la période post-quantique. En décembre 2016, NIST (National Institute of Standards and Technology of US) [Nist] a lancé un appel à propositions de schémas à clefs publiques post-quantiques. Quand on parle de cryptographie post-quantique, on suppose que l'adversaire a accès à une machine quantique.

1.2 Organisation de cette thèse

Dans le présent chapitre, nous faisons une introduction de la cryptographie et le résumé de nos contributions. Le reste de cette thèse est organisé en trois parties :

1. ❦ **Partie I** : Cette partie regroupe nos contributions sur les courbes elliptiques et hyperelliptiques. Elle se subdivise en trois chapitres.
   — Dans le chapitre 2 (extrait d'une publication [SBDK17]), nous construisons deux fonctions d'encodage sur de nouvelles familles de courbes hyperelliptiques de genre $g = 2$. Nous montrons aussi que nos encodages sont bien distribués en utilisant les travaux de Farashahi et al. [FFS+13] et donc ils peuvent être utilisés pour construire des fonctions de hachage d'un corps fini vers le Jacobien de courbes hyperelliptiques de genre $g = 2$.
   — Dans le chapitre 3 (extrait d'une publication [SD18]), nous généralisons un de nos encodages en genre 2 en construisant, d'une part, quatre nouvelles fonctions d'encodage en genre $g = 1, 3, 4, 5$ et d'autre part, une fonction d'encodage générale qui unifie nos quatre nouvelles fonctions d'encodage et celle en genre 2.
   — Dans le chapitre 4 (extrait d'une publication [DSS18]), nous faisons l'état de l'art des fonctions d'encodage et de hachage sur les courbes elliptiques et hyperelliptiques et les techniques de construction utilisées. Nous montrons que notre fonction d'encodage générale est bien distribuée en utilisant le même framework de Farashahi et al. [FFS+13].
2. ❦**Partie II** : Dans cette partie, au chapitre 5, nous décrivons SimulaMath (`https://simulamath.org`), qui est un nouveau logiciel de simulation et de calcul dédié à l'enseignement des mathématiques et à la recherche que nous avons développé avec le langage python. On y décrit SimulaMath et ses applications (son utilisation) sur les courbes elliptiques, les réseaux arithmétiques, les codes linéaires, l'algèbre linéaire, l'analyse, la résolution d'équations et de systèmes, la statistique descriptive, les probabilités, les diagrammes, les graphiques à deux et trois dimensions.
3. ❦ **Partie III** : **Annexes** : Enfin dans cette partie, nous faisons quelques rappels.
   — Dans l'annexe A, nous rappelons quelques définitions et propriétés sur les groupes, anneaux et corps et extensions de corps. Nous rappelons aussi quelques propriétés sur les caractères, les symboles de Legendre, de Gauss et de Jacobi sans oublier les algorithmes classiques de calcul d'une racine carrée dans un corps fini $\mathbb{F}_q$.
   — Nous rappelons, dans l'annexe B, les variétés affines et projectives, quelques propriétés et algorithmes sur les différentes familles d'une courbes elliptiques. Nous rappelons aussi quelques propriétés des isogénies entre courbes elliptiques

et l'algorithme de Vélu [Vel71]. Nous rappelons également la notion de diviseurs d'une courbe algébrique et spécialement la notion de diviseur d'une courbe elliptique et hyperelliptique.

1.3 Résumé des travaux

Dans cette section, nous faisons un résumé de nos différentes contributions.

1.3.1 Résumé du chapitre 2

Dans le chapitre 2, nous généralisons la méthode Elligator 2 de Bernstein et al. [BHKL13] sur les courbes elliptiques. Nous proposons d'abord, dans la section 2.3, deux nouvelles fonctions d'encodage *déterministes et presque injectives* en genre $g = 2$ sur $\mathbb{F}_q$, l'une sur la courbe hyperelliptique donnée par l'équation $\mathbb{H}^1 : y^2 = F_1(x) = x^5 + bx^3 + dx + e$, avec $b = sw^2$, $d = \dfrac{sw^4}{2}$, $e = \dfrac{s-10}{10}w^5$ et l'autre par l'équation $\mathbb{H}^2 : y^2 = F_2(x) = x^5 + ax^4 + e$ avec $a \in \mathbb{F}_q^*$ et $e = 4aw^4$.

— Dans la sous-section 2.3.1, nous définissons la fonction $\phi_1 : \mathcal{R}_1 \ni r \mapsto \phi_1(r) = (x, y)$ où $\mathcal{R}_1$ est un sous-ensemble de $\mathbb{F}_q$ tel que $\mathbb{F}_q \backslash \mathcal{R}_1$ contient au plus 11 éléments et x, y sont donnés par l'algorithme suivant (Algorithme 1). Nous montrons que $\phi_1(r) = (x, y)$ est bel et bien un point de la courbe hyperelliptique $\mathbb{H}^1$ i.e $(x, y) \in \mathbb{H}^1$.

Algorithme 1 : Hashing-1-Genus2-Encoding

Entrées : La courbe hyperelliptique $\mathbb{H}^1$, un élément $r \in \mathcal{R}_1$
Sortie : Un point (x, y) de $\mathbb{H}^1$

1 $v := w[ur^2(-50 - 35s) - 1]$;
2 $\varepsilon := \chi(v^5 + v^3 + dv + e)$;
3 $x := \dfrac{1+\varepsilon}{2}v + \dfrac{1-\varepsilon}{2}\left(\dfrac{w(-v+w)}{v+w}\right)$;
4 $y := -\varepsilon\sqrt{x^5 + bx^3 + dx + e}$;
5 **Retourner** (x, y).

Nous montrons également dans quelle condition un point $(x, y) \in \mathbb{H}^1$ admet un antécédent par la fonction ϕ_1 et si tel est le cas nous déterminons explicitement tous les antécédents qui sont exactement au nombre de 2.

— Dans la sous-section 2.3.2 nous construisons la fonction d'encodage $\phi_2 : \mathcal{R}_2 \to \mathbb{H}^2$:

$r \mapsto \phi_2(r) = (x, y)$.

Algorithme 2 : Hashing-2-Genus2-Encoding

Entrées : la courbe hyperelliptique $\mathbb{H}^2$, un élément $r \in \mathcal{R}_2$.
Sortie : un point (x, y) on $\mathbb{H}^2$

1 $v = v(r^2) := w(1 - ur^2)$;
2 $\varepsilon := \chi(v^5 + av^4 + 4aw^4)$;
3 $x := \dfrac{1+\varepsilon}{2} v + \dfrac{1-\varepsilon}{2}\left(\dfrac{-wv}{v-w}\right)$;
4 $y := -\varepsilon\sqrt{x^5 + ax^4 + 4aw^4}$;
5 **Retourner** (x, y).

En utilisant les résultats de Farashahi et al. [FFS+13], nous faisons la preuve du théorème suivant dans la section 2.4.

Théorème 1.3.1
On suppose $q = 3 \bmod 4$. *Alors pour tout caractère non trivial* χ *de* $\mathbb{J}_1(\mathbb{F}_q)$, *on a*

$$\left|\sum_{t\in\mathbb{F}_q} \chi_q(\phi_1(t))\right| \leqslant 32\sqrt{q} + 119$$

Ce qui montre que notre fonction d'encodage ϕ_1 est bien distribuée d'après Farashahi et al. [FFS+13] et donc la fonction de hachage $H : \mathbb{F}_q^s \to \mathbb{J}_1(\mathbb{F}_q) : m \mapsto H(m) = h_1(\phi_1(x)) + \ldots + h_s(\phi_1(x)), s > 2$ est indifferentiable à un oracle aléatoire si $h_1, \ldots, h_s$ sont des fonctions de hachage classiques modélisées comme des oracles aléatoires .

1.3.2 Résumé du chapitre 3

Après avoir réussi à encoder en genre 2 sur la courbe hyperelliptique donnée par l'équation $\mathbb{H}^1 : y^2 = F_1(x) = x^5 + bx^3 + dx + e$ en généralisant la méthode Elligator 2 de Bernstein et al. [BHKL13] initialement pour le genre 1, nous avons essayé

— en premier lieu, d'étendre notre technique d'encodage en genre $g \geqslant 3$ et en genre 1 sur le même type de famille de courbes hyperelliptiques $\mathbb{H}_g : y^2 = f_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0$;
— et en deuxième lieu, de trouver des formules unifiées qui regroupent nos différents encodages.

Dans la section 3.2, nous proposons nos quatre encodages déterministes et presque injectifs $\psi_i : \mathcal{R}_i \subset \mathbb{F}_q \to \mathbb{H}_i$ pour $i = 1, 3, 4, 5$, où :

$$\mathbb{H}_1 : y^2 = f_1(x) = x^3 + a_1x + a_0;$$

$$\mathbb{H}_3 : y^2 = f_3(x) = x^7 + a_5x^5 + a_3x^3 + a_1x + a_0;$$

$$\mathbb{H}_4 : y^2 = f_4(x) = x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0;$$

$$\mathbb{H}_5 : y^2 = f_5(x) = x^{11} + a_9x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0.$$

Dans la section 3.3, nous présentons un encodage $\psi_g : \mathcal{R}_g \subset \mathbb{F}_q \to \mathbb{H}_g$ déterministe et presque injectif qui unifie nos nouveaux encodages en genre $g = 1, 3, 4, 5$ et celui en

genre 2. Pour tout $r \in \mathcal{R}_g$, son image $\psi_g(r) = (x, y) \in \mathbb{H}_g$ où $\mathbb{H}_g : y^2 = h_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0$ et (x, y) est donné par l'algorithme suivant.

Algorithme 3 : Genus-g-Encoding6-Generalization

Entrées : la courbe hyperelliptique $\mathbb{H}_g$, un élément $r \in \mathcal{R}_g$
Sortie : un point (x, y) on $\mathbb{H}_g$

1 $v := v(g) = w[ur^2(-m_g s - n_g) - 1]$;
`/* ` m_g `et` n_g `sont des fonctions qui ne dépendent que du genre` g `*/`
2 $\varepsilon := \chi(v^{(2g+1)} + a_{(2g-1)}v^{(2g-1)} + a_{(2g-3)}v^{(2g-3)} + \ldots + a_1v + a_0)$;
3 $x := \dfrac{1+\varepsilon}{2}v + \dfrac{1-\varepsilon}{2}\left(\dfrac{w(-v+w)}{v+w}\right)$;
4 $y := -\varepsilon\sqrt{x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0}$;
5 **Retourner** (x, y).

Dans la section 3.6, nous proposons une implémentation de notre fonction d'encodage unifiée en utilisant le logiciel SageMath [Dev17]. Le code est aussi disponible sur Github [Sec18].

1.3.3 Résumé du chapitre 4

Au chapitre 4, nous faisons l'état de l'art des différents encodages et hachages sur les courbes elliptiques et hyperelliptiques existants dans la littérature. Nous montrons également que la fonction d'encodage générale définie au chapitre 3 est bien distribuée.

— Dans la section 4.2, nous faisons un résumé des encodages existants.

TABLE 1.1 – Classification des fonctions d'encodage sur les courbes elliptiques et hyperelliptiques

Encodage	Courbe	$\mathbb{F}_q$	Méthode SWU	Méthode Icart	Méthode Injective	Méthode Elligator-2
Brier *et al.* [BCI+10]	$y^2 = x^3 + a + b$	$q \equiv 2 \bmod 3$	✓			
Fouque et al.[FT12]	$y^2 = x^3 + b$	$q \equiv 1 \bmod 3$	✓			
Farashahi [Far11]	$x^3 + y^3 + 1 = 3dxy$	$q \equiv 2 \bmod 3$		✓		
Kammerer *et al.* [KLR10]	$x^3 + y^3 + 1 = 3dxy$	$q \equiv 2 \bmod 3$		✓		
Bernstein *et al.* [BHKL13]	$x^2 + y^2 = 1 + d^2y^2$	$q \equiv 3 \bmod 4$			✓	
Diarra *et al.* [DDK17]	$x^2 + y^2 = 1 + dx^2y^2$	$q \neq 2^n$				✓
He *et al.* [HYW15]	$ax(y^2 - c) = by(x^2 - d)$	$q \equiv 3 \bmod 4$	✓			
	$ax(y^2 - c) = by(x^2 - d)$	$q \equiv 2 \bmod 3$		✓		
Diarra *et al.* [DDK17]	$x(ay^2 - 1) = y(bx^2 - 1)$	$q \equiv 3 \bmod 4$			✓	
	$\alpha x(y^2 - 1) = \beta y(x^2 - 1)$	$q \neq 2^n$				✓

TABLE 1.1 – Classification des fonctions d'encodage sur les courbes elliptiques et hyperelliptiques

Encodage	Courbe	$\mathbb{F}_q$	Méthode SWU	Méthode Icart	Méthode Injective	Méthode Elligator-2
Yu *et al.* [YWLT13]	$by^2 = x^3 + ax^2 + x$	$q \equiv 2 \bmod 3$		✓		
		$q \equiv 3 \bmod 4$	✓			
Yu *et al.* [YWL+15]	$y^2 = x^4 + 2bx^2 + 1$	$q \equiv 2 \bmod 3$		✓		
		$q \equiv 3 \bmod 4$	✓			
He *et al.* [HYW17]	$au^2 + v^2 = bv^2 + w^2 = 1$	$q \equiv 2 \bmod 3$		✓		
		$q \equiv 3 \bmod 4$	✓			
Ulas [Ula07]	$y^2 = x^n + ax + b$		✓			
	$y^2 = x^n + ax^2 + bx$					
Kammerer *et al.* [KLR10]	$y^2 = (x^3 + 3ax + 2)^2 + 8b^3$	$q \equiv 2 \bmod 3$		✓		
	$y^2 = \lambda((x^3 + 3\mu x + 2a)^2 + 4b)$	$q \equiv 2 \bmod 3$		✓		
Dans cette thèse	$y^2 = x^5 + bx^3 + dx + e$	$q \equiv 7 \bmod 8$				✓
Seck *et al.* [SBDK17]	$y^2 = x^5 + ax^4 + e$	$q = 5^n$				✓
	$y^2 = x^5 + ax^4 + cx^2 + dx$	$q \equiv 3 \bmod 4$				✓
Dans cette thèse	$y^2 = F_g(x),\ g \leqslant 5$	$q \equiv 7 \bmod 8$				✓
Seck et Diarra [SD18]						

— La section 4.3 est dédiée aux différentes constructions de fonctions de hachage indifférentiables à un oracle aléatoire sur le jacobien d'une courbe elliptique ou hyperelliptique et les techniques utilisées.

TABLE 1.2 – Fonctions de hachage indifférentiable sur les courbes (hyper)elliptique existantes

Hachage sur la courbe	Méthode de hachage	Encodage utilisé
$E_b :\ y^2 = x^3 + b$	$H = f \circ h$	Boneh et Franklin [BF01]
$E_b :\ y^2 = x^3 + b$	$H = f \circ h_1 + \ldots + f \circ h_s$	Fouque et Tibouchi [FT12]
$E_d :\ x^2 + y^2 = 1 + dx^2y^2$	$H = f \circ h_1 + \ldots + f \circ h_s$	Diarra et al. [DDK17]
$E_{a,b} : y^2 = x^3 + ax + b$	$H = f \circ h_1 + h_2\mathbb{G}$	Brier et al. [BCI+10]
	$H = f \circ h_1 + h_2\mathbb{G}$	
$E_{a,b} = y^2 = x^3 + ax + b$	$H = f \circ h_1 + \ldots + f \circ h_s$	Farashahi et al. [FFS+13]
$H = x^3 + (y + c)(3x + 2a + \frac{2b}{y}) = 0$	$H = f \circ h_1 + \ldots + f \circ h_s$	Farashahi et al. [FFS+13]
$E_1 :\ ax(y^2 - c) = by(x^2 - d)$	$H = f \circ h_1 + h_2\mathbb{G}$	He et al. [HYW15]
$E_b = y^2 = x^4 + 2bx^2 + 1$	$H = f \circ h_1 + \ldots + f \circ h_s$	Yu et al. [YWL+15]

TABLE 1.2 – Fonctions de hachage indifférentiable sur les courbes (hyper)elliptique existantes

Hachage sur la courbe	Méthode de hachage	Encodage utilisé
$E_{a,b} : by^2 = x^3 + ax^2 + x$	$H = f \circ h_1 + h_2\mathbb{G}$ $H = f \circ h_1 + \ldots + f \circ h_s$	Yu et al. [YWLT13]
$E_{a,b} : au^2 + v^2 = bv^2 + w^2 = 1$	$H = f \circ h_1 + \ldots + f \circ h_s$	He et al. [HYW17]
$\mathbb{H}^1 : y^2 = x^5 + ax^4 + cx^2 + dx$	$H = f \circ h_1 + \ldots + f \circ h_s$	Seck *et al.* [SBDK17]
$\mathbb{H}^2 : y^2 = x^5 + a_3x^3 + a_1x + a_0$	$H = f \circ h_1 + \ldots + f \circ h_s$	Dans cette thèse [SBDK17]
$\mathbb{H}_g : y^2 = x^{2g+1} + a_{2g-1}x^{2g-1} + \ldots + a_1x + a_0$	$H = f \circ h_1 + \ldots + f \circ h_s$	Dans cette thèse [SD18]

— Dans la section 4.4, nous montrons que la fonction d'encodage ψ_g unifiant nos différents encodages est bien distribuée en utilisant les travaux de Farashahi et al. [FFS+13]. Nous prouvons le théorème suivant :

Théorème 1.3.2
Pour tout caractère non trivial χ de $\mathbb{H}_g(\mathbb{F}_q)$, la somme de caractères $S_{\psi_g}(\chi)$ satisfait :

$$\left|S_{\psi_g}(\chi)\right| \leqslant (16g)\sqrt{q} + (44g + 31)$$

où g est le genre de la courbe hyperelliptique $\mathbb{H}_g$ et $S_{\psi_g}(\chi) = \sum_{r\in\mathbb{F}_q} \chi(\psi_g(r))$.

1.3.4 Résumé du chapitre 5

Au chapitre 5, nous décrivons SimulaMath qui est un logiciel de calcul pour la recherche, l'enseignement et l'apprentissage en mathématiques. Il faut noter que toutes les figures (graphiques) dans cette thèse sont réalisées avec ce logiciel.

1.3.5 Publications et Preprints

Liste des publications

① Michel Seck, Hortense Boudjou, Nafissatou Diarra and Ahmed Youssef Ould Cheikh, "On Indifferentiable Hashing into the Jacobian of Hyperelliptic Curves of Genus 2", Springer Lecture Notes in Computer Science (LNCS), In M. Joye and A. Nitaj, editors, Advances in Cryptology AFRICACRYPT-2017, 205-222(2017)

② Seck M., Diarra N. : Unified Formulas for Some Deterministic Almost-Injective Encodings into Hyperelliptic Curves. In : Joux A., Nitaj A., Rachidi T. (eds) Progress in Cryptology - AFRICACRYPT 2018. AFRICACRYPT 2018. Lecture Notes in Computer Science, vol 10831. Springer, Cham (2018)

③ Diarra N., Seck M. and Sow D.. A Note on Encoding and Hashing into Elliptic and Hyperelliptic Curves . In A Collection of Papers in Mathematics and Related Sciences, a festschrift in honour of the late Galaye Dia (Editors : Seydi H., Lo G.S.

and Diakhaby A.). Spas Editions, Euclid Series Book, pp. 565 –593. (2018) Doi : 10.16929/sbs/2018.100-07-01

Preprints

① Seck, M. : SimulaMath : A software for teaching, learning and research in mathematics. https://simulamath.org (2018)

Première partie

Hachage sur les courbes hyperelliptiques

Chapitre 2

Sur le hachage indifférentiable sur la Jacobienne de courbes hyperelliptiques de genre 2 [1]

Contents

1. Ce chapitre est basé sur l'article publié suivant :

Michel Seck, Hortense Boudjou, Nafissatou Diarra and Ahmed Youssef Ould Cheikh, "On Indifferentiable Hashing into the Jacobian of Hyperelliptic Curves of Genus 2", Springer Lecture Notes in Computer Science (LNCS), In M. Joye and A. Nitaj, editors, Advances in Cryptology AFRICACRYPT-2017, 205-222(2017)

De nombreux auteurs ont étudié le problème de la construction de fonctions de hachage indifférentiables et déterministes sur des courbes elliptiques et hyperelliptiques en utilisant des encodages bien distribués. Dans ce chapitre, nous proposons deux fonctions d'encodage qui sont compatibles avec le hachage indifférentiable sur deux familles de courbes hyperelliptiques du genre 2 : $\mathbb{H}^1 : y^2 = F_1(x) = x^5 + bx^3 + dx + e$;
$\mathbb{H}^2 : y^2 = F_2(x) = x^5 + ax^4 + e$. Nous montrons que nos encodages peuvent être utilisés pour construire des fonctions de hachage indifférentiables dans la Jacobienne des courbes hyperelliptiques $\mathbb{H}^1$ et $\mathbb{H}^2$, en utilisant une technique développée par Farashahi `et al.` en 2013 (J. Math. Comput). Ces nouveaux encodages ont la même complexité asymptotique à savoir $\mathcal{O}(\log^{2+\circ(1)} q)$ (voir Kammerer *et al.*, Pairing 2010).

2.1 Introduction

La cryptographie basée sur les courbes elliptiques (ECC) offre plusieurs services qui combinent à la fois vitesse, sécurité et faible consommation de mémoire. Elle est donc de plus en plus utilisée par les concepteurs de protocoles cryptographiques. Parmi ces protocoles, on peut citer entre autres, les schémas de chiffrement basés sur l'identité. Généralement, dans de tels schémas, l'identité de l'utilisateur est le plus souvent associée à un point de la courbe (hyper)elliptique, comme dans le cas du schéma de Boneh et Franklin [BF01] pour le cas particulier des courbes supersingulières. De plus, lorsqu'on utilise un protocole qui nécessite une fonction de hachage, il faut pouvoir coder sur le groupe attaché à la courbe, à savoir le groupe de points d'une courbe elliptique ou le Jacobien d'une courbe hyperelliptique. Pour le cas des courbes elliptiques, il existe de nombreux encodages déterministes [SvdW06, Ica09, Far14, FJT13, BHKL13, Ham15] et des constructions de fonctions de hachage basées sur des encodages bien distribués [BCI+10, FFS+13, FT10, Tib11].

En 2007, Ulas [Ula07] a construit des encodages déterministes pour la famille de courbes hyperelliptiques de la forme $y^2 = x^n + ax + b$ ou $y^2 = x^n + ax^2 + bx$, où $n \geqslant 5$. Les encodages d'Ulas sont basés sur l'égalité de Skalba et nécessitent de calculer une racine carrée dans $\mathbb{F}_q$. Une version simplifiée de l'encodage d'Ulas a été proposée plus tard par Brier *et al.* dans [BCI+10] à CRYPTO 2010, où les auteurs expliquent aussi comment construire des fonctions de hachage indifférentiables d'oracles aléatoires et basées sur des encodages déterministes sur des courbes elliptiques (comme la fonction d'Icart [Ica09] ou l'algorithme de SWU[Ula07]).

En 2010, Kammerer et *al.* [KLR10] ont proposé de nouveaux encodages pour quelques modèles de courbes hyperelliptiques, dont la courbe $y^2 = (x^3 + 3ax + 2)^2 + 8bx^3$ sur $\mathbb{F}_q$, avec $q \equiv 2 \bmod 3$. Farashahi *et al.*[FFS+13] ont aussi proposé une autre technique basée sur les sommes de caractères qui leur a permis d'établir l'indifférentiabilité de fonctions de hachage (qui sont basées sur l'existence d'encodages déterministes bien distribués dans la Jacobienne de courbes hyperelliptiques). Ils ont aussi appliqué leur technique à la courbe hyperelliptique proposée par Kammerer et *al.* [KLR10]. Pour le cas des courbes hyperelliptiques impaires (qui sont des courbes définies par $y^2 = f(x)$ avec $f(-x) = -f(x)$), Fouque et Tibouchi utilisant la technique de Farashahi *et al.* [FFS+13], ont proposé une méthode explicite pour obtenir des éléments de la Jacobienne, à l'aide d'encodages injectifs bien distribués sur les courbes hyperelliptiques considérées.

Contributions : Il est bien connu que la cryptographie basée sur les courbes hyperelliptiques est fortement liée aux calculs sur la jacobienne de la courbe sous-jacente. Et pour cela, il faut pouvoir calculer un point de la courbe en temps polynomial et ce, de façon déterministe. Notre principale contribution dans ce chapitre est la construction d'encodages déterministes pour deux familles de courbes hyperelliptiques qui n'ont pas été couvertes par les travaux précédents. De plus, nous montrons que chacun de ces encodages est $2:1$ [2], inversible sous certaines conditions et peut être étendu à l'ensemble $\mathbb{F}_q$ tout entier. Et en utilisant des résultats de Farashahi *et al.*[FFS+13], nous montrons comment on peut utiliser ces nouveaux encodages pour construire des fonctions de hachage indifférentiables sur la Jacobienne des courbes hyperelliptiques $\mathbb{H}_i,\ i = 1, 2$. Nos encodages peuvent être calculés en $\mathcal{O}(\log^{2+\circ(1)} q)$ opérations sur $\mathbb{F}_q$.

Organisation du chapitre : ce chapitre est organisé comme suit :
- Dans la section 2.2, nous donnons quelques préliminaires utiles sur les racines carrées dans les corps finis.
- Dans la section 2.3, nous présentons nos nouveaux encodages sur les courbes hyperelliptiques $H_i,\ i = 1, 2$. Nous donnons aussi les algorithmes pour inverser ces encodages que nous appelons *algorithmes de décodage*.
- Et enfin, dans la section 2.4, nous commençons par rappeler quelques définitions et résultats relatifs aux *encodages bien distribués*. Et finalement, nous montrons que nos nouveaux encodages déterministes peuvent être utilisés pour construire des fonctions de hachage indifférentiables.

2.2 Préliminaires

Nous rappelons ici quelques définitions et propriétés sur les sommes de caractères, les caractères quadratiques, les courbes (hyper)elliptiques, la formule de Riemann-Hurwitz et les fonctions d'encodage sur les courbes. Pour plus d'informations sur ces concepts, voir la partie III des annexes.

2.2.1 Caractères et racine carrée sur les groupes abéliens finis

Définition 2.2.1.

Soit G un groupe abélien fini (noté multiplicativement). Un caractère sur G est un homomorphisme de groupes $\chi : G \to \mathbb{C}^*$, où $\mathbb{C}^*$ est le groupe multiplicatif des nombres complexes non nuls.

- Pour un caractère χ sur G, nous avons $\chi(g_1 g_2) = \chi(g_1)\chi(g_2)$ et $\chi(1) = 1$.
- Le caractère trivial 1_G est la fonction sur G où $1_G(g) = 1$ pour tout $g \in G$.
- L'ensemble des caractères sur G muni de la multiplication $(\chi_1\chi_2)(g) = \chi_1(g)\chi_2(g)$ est un groupe abélien appelé groupe de caractères de G, et est noté par $\bar{G}$.

2. Cela signifie que tout point de la courbe hyperelliptique appartenant à l'image de l'encodage a exactement deux antécédents.

Caractère quadratique et racine carrée

Soit $q = p^n$ avec $p \geqslant 3$ un nombre premier, et notons par $\mathbb{F}_q$ le corps fini qui a q éléments.

1. Le **caractère quadratique** (ou symbole de Legendre généralisé) est la fonction χ_q définie par $\chi_q : \mathbb{F}_q \to \mathbb{F}_q : u \mapsto \chi_q(u) = u^{(q-1)/2}$ et vérifiant : $\chi_q(u) = 1$ si u est un carré non nul ; $\chi_q(u) = -1$ si u n'est pas un carré ; et $\chi_q(u) = 0$ si $u = 0$. Les propriétés suivantes sont également vérifiées : $\chi_q(uv) = \chi_q(u) \cdot \chi_q(v)$ pour tous $u, v \in \mathbb{F}_q$ et si $q \equiv 3 \bmod 4$, $\chi_q(-1) = -1$, $\chi_q(\chi_q(u)) = \chi_q(u)$, pour tout $u \in \mathbb{F}_q$. Si $q \equiv 1 \bmod 4$, alors $\chi_q(-1) = 1$.
2. **Racine carrée :** Soit $\mathbb{F}_q^2 = \{a^2 : \ a \in \mathbb{F}_q\}$. Une fonction racine carrée $\sqrt{\ }$ sur $\mathbb{F}_q$ est définie par : $\sqrt{\ } : \ \mathbb{F}_q^2 \to \mathbb{F}_q : a^2 \mapsto \sqrt{a^2} \in \{a, -a\}$. Dans le cas où $q \equiv 3 \bmod 4$, le nombre $a^{\frac{q+1}{4}}$ est appelé la racine carrée principale de a ; on peut donc prendre la racine carrée principale comme une fonction racine carrée. Si $q \geqslant 3$, q premier, on peut prendre $\sqrt{\mathbb{F}_q^2} = \{0, 1, \ldots, \frac{q-1}{2}\}$.

2.2.2 Formule de Riemann Hurwitz

Nous rappelons ici la formule de Riemann Hurwitz qui relie la ramification d'un morphisme séparable à son degré, ainsi que le genre d'une courbe.

Théorème 2.2.2
Soit $\phi : X \to Y$ un morphisme séparable de courbes, de degré d. Soit g_X et g_Y le genre de X et Y respectivement. Nous avons alors $2g_X - 2 \geqslant d(2g_Y - 2) + \sum_{P \in X}(e_P - 1)$ avec égalité si et seulement si ϕ est ramifié et sa ramification est "tame". Noter que e_P indique l'indice de ramification en P.

2.2.3 Encodages sur les courbes (hyper)elliptiques

Définition 2.2.3.

Étant donné une courbe hyperelliptique $\mathbb{H}$ sur $\mathbb{F}_q$, un encodage sur $\mathbb{H}$ est une fonction $f : \mathbb{F}_q \to \mathbb{H}$.

Les encodages utilisés en cryptographie basée sur les courbes (hyper)elliptiques doivent vérifier certaines propriétés intéressantes, comme *l'admissibilité* ou *la presque-injectivité*. Nous donnons ici des définitions plus formelles de telles propriétés (que l'on peut trouver dans [FFS+13, BHKL13, DDK17, SD18]).

Définition 2.2.4 (Encodage presque-injectif)**.**

Un encodage $f : \mathbb{F}_q \to \mathbb{H}$ est dit *presque-injectif* s'il existe $S \subset \mathbb{F}_q$ tel que :
(i) $S \cap (-S) = \{0\}$;
(ii) $f(r) = f(-r), \forall r \in \mathbb{F}_q$;
(iii) $\#f^{-1}(f(r)) = 2, \forall r \in \mathbb{F}_q^*$.
Ainsi, f est presque injectif si sa restriction à un certain sous-ensemble S de $\mathbb{F}_q$ est injective.

Définition 2.2.5 (encodage admissible [BCI+10])**.**

Un encodage $f : \mathbb{F}_q \to \mathbb{H}$ est *ϵ-admissible* s'il satisfait les propriétés suivantes :
(i) calculable : f est calculable en temps polynomial de façon déterministe.
(ii) régulier : pour r uniformément distribué dans $\mathbb{F}_q$, la distribution de $f(r)$ est ϵ-statistiquement indistinguable de la distribution uniforme dans $\mathbb{H}$.
(iii) samplable : il existe un algorithme randomisé efficace $I : \mathbb{H} \to \mathbb{F}_q$ tel que pour tout $P \in \mathbb{H}$, $I(P)$ induit une distribution qui est statistiquement ϵ-indistinguable de la distribution uniforme dans $f^{-1}(P)$.
Et f est un encodage *admissible* si ϵ est une fonction négligeable du paramètre de sécurité.

Définition 2.2.6 (weak-encodage [BCI+10])**.**

On dit qu'une fonction $f : \mathbb{F}_q \to \mathbb{H}$ est un *α-weak-encodage* si elle satisfait les propriétés suivantes :
(i) calculable : f est calculable en temps polynomial de manière déterministe ;
(ii) α-bornée : Pour r uniformément distribué dans $\mathbb{F}_q$, la distribution de $f(r)$ est α-bornée dans $\mathbb{H}$, c'est-à-dire que l'inégalité $Pr[f(r) = P] \leqslant \frac{\alpha}{\#\mathbb{H}}$ est satisfaite pour tout $P \in \mathbb{H}$;
(iii) samplable : il existe un algorithme randomisé efficace I tel que $I(P)$ induit la distribution uniforme dans $f^{-1}(P)$ pour tout $P \in \mathbb{H}$. De plus, $I(P)$ renvoie $N_P = \#f^{-1}(P)$.
Et f est un weak-encodage si α est une fonction polynomiale du paramètre de sécurité.

Les weak-encodages sont essentiellement ceux dont la taille de l'ensemble image est une fraction constante positive de la taille de la courbe, ce qui inclut tous les encodages connus sur les courbes (hyper)elliptiques.

Définition 2.2.7 (encodage bien-distribué [FFS+13]).

Soit C une courbe projective lisse sur un corps fini $\mathbb{F}_q$, $\mathbb{J}\mathrm{ac}$ sa Jacobienne, f une fonction $\mathbb{F}_q \to C(\mathbb{F}_q)$ et B une constante positive. On dit que f est B-bien distribuée si pour tout caractère non trivial χ de $\mathbb{J}\mathrm{ac}(\mathbb{F}_q)$, on a :

$$|S_f(\chi)| \leqslant B\sqrt{q} \text{ où } S_f(\chi) = \sum_{u \in \mathbb{F}_q} \chi(f(u))$$

f est bien distribuée s'il est B-bien distribuée pour un certain B borné indépendamment du paramètre de sécurité.

Considérons un encodage f sur une courbe C, et notons par $\mathbb{J}\mathrm{ac}$ la Jacobienne de C. Nous définissons la fonction $f^{\otimes s} : (\mathbb{F}_q)^s \to \mathbb{J}\mathrm{ac}(\mathbb{F}_q)$ comme suit :

$$f^{\otimes s}(u_1, u_2, \ldots, u_s) = f(u_1) + f(u_2) + \ldots + f(u_s), \forall (u_1, u_2, \ldots, u_s) \in (\mathbb{F}_q)^s$$

Théorème 2.2.8 ([FFS+13])
Soit $h : \tilde{C} \to C$ un morphisme non constant de courbes, et χ un caractère non trivial de $\mathbb{J}\mathrm{ac}(\mathbb{F}_q)$, où $\mathbb{J}\mathrm{ac}$ est la Jacobienne de C. Supposons que h ne se factorise pas via un morphisme non ramifié $Z \to X$. Alors :

$$\left| \sum_{P \in \tilde{C}(\mathbb{F}_q)} \chi(h(P)) \right| \leqslant (2\tilde{g} - 2)\sqrt{q}$$

où $\tilde{g}$ est le genre de $\tilde{C}$.
De plus, si q est impair et ϕ est une fonction rationnelle non constante sur $\tilde{C}$, on a alors

$$\left| \sum_{P \in \tilde{C}(\mathbb{F}_q)} \chi(h(P)) \left(\frac{\phi(P)}{q} \right) \right| \leqslant (2\tilde{g} - 2 + \deg(\phi))\sqrt{q} \tag{2.1}$$

où $\left(\frac{\cdot}{q}\right)$ indique le symbole de Legendre (généralisé).

2.3 Encodages presque injectifs et inversibles sur deux familles de courbes hyperelliptiques

Dans cette section, nous proposons deux encodages *presque-injectifs* sur les courbes hyperelliptiques en utilisant la technique de Elligator [BHKL13] pour les courbes elliptiques. Tous nos encodages sont *presque-injectifs* dans le sens de la définition 2.2.4.

A notre connaissance, les résultats présentés dans ce chapitre ne sont pas couverts par les travaux précédents (résumés dans l'introduction). Notons que dans [FFS+13], l'encodage est valable pour les courbes hyperelliptiques impaires. Dans [KLR10], [Ula07],

[BCI+10], les encodages ne sont pas presque-injectifs. Tous les encodages que nous proposons ici sont presque-injectifs et peuvent être utilisés pour des courbes hyperelliptiques non-impaires. De plus, dans la section 2.4, nous prouvons que nos encodages sont bien distribués ; ils peuvent donc être utilisés pour construire des fonctions de hachage indifférentiables.

2.3.1 Encoding presque-injectif sur $\mathbb{H}^1$

Soit $\mathbb{F}_q$ un corps fini avec $\text{char}(\mathbb{F}_q) = p \neq 2, 5$, $q = p^n$ avec $p \geqslant 3$ un nombre premier. Nous supposons que $q \equiv 1 \bmod 8$ ou $q \equiv 7 \bmod 8$, alors 2 est un carré dans $\mathbb{F}_q$.

Soit $s \in \mathbb{F}_q^*$ tel que $7s^2 + 20s - 100 = 0$:

— Si $p \neq 7$ et "$q \equiv 1 \bmod 8$ ou $q \equiv 7 \bmod 8$", on a $\Delta_s = 3200 = 2 \times 4^2 10^2$ qui est un carré, alors $s = \dfrac{-10 \pm 20\sqrt{2}}{7}$.

— Si $p = 7$ et "$q \equiv 1 \bmod 8$ ou $q \equiv 7 \bmod 8$", alors $s = 5$.

Soit $w \in \mathbb{F}_q^*$ un paramètre arbitraire.

Soit $\mathbb{H}^1 : y^2 = F_1(x) = x^5 + bx^3 + dx + e$, avec $b = sw^2$, $d = \dfrac{sw^4}{2}$, $e = \dfrac{s-10}{10}w^5$ une courbe hyperelliptique de genre 2 sur $\mathbb{F}_q$ avec la condition antérieure sur q.

Soit u un paramètre tel que $\chi(u) = -1$ et posons

$$\mathcal{R}_1 = \left\{ r \in \mathbb{F}_q^*, [ur^2(-50-35s)-1]^5 + s[ur^2(-50-35s)-1]^3 + \frac{s}{2}ur^2(-50-35s) \neq \frac{2s+5}{5} \right\}$$

Algorithme 4 : Hashing-2-Genus2-Encoding

Entrées : La courbe hyperelliptique $\mathbb{H}^1$, un élément $r \in \mathcal{R}_1$
Sortie : Un point (x, y) sur $\mathbb{H}^1$

1 $v := w[ur^2(-50-35s)-1]$;
2 $\varepsilon := \chi(v^5 + v^3 + dv + e)$;
3 $x := \dfrac{1+\varepsilon}{2}v + \dfrac{1-\varepsilon}{2}\left(\dfrac{w(-v+w)}{v+w}\right)$;
4 $y := -\varepsilon\sqrt{x^5 + bx^3 + dx + e}$;
5 **Retourner** (x, y).

Définition 2.3.9.

Dans la situation de l'algorithme 4, l'encodage pour la courbe hyperelliptique $\mathbb{H}^1$ est la fonction $\phi_1 : \mathcal{R}_1 \to \mathbb{H}^1 : r \mapsto \phi_1(r) = (x, y)$.

Théorème 2.3.10
L'algorithme 4 calcule un encodage presque-injectif de façon déterministe $\phi_1 : \mathcal{R}_1 \to \mathbb{H}^1 : r \mapsto \phi_1(r) = (x, y)$, en temps $\mathcal{O}(log^{2+\circ(1)}q)$, où $\mathcal{S}_1 = \mathbb{F}_q \backslash \mathcal{R}_1$ est un sous-ensemble de $\mathbb{F}_q$ d'au plus 11 éléments.

Preuve

1. Puisque $v = w[ur^2(-50-35s)-1]$ on a $v^5+bv^3+dv+e = w^5[ur^2(-50-35s)-1]^5+bw^3[ur^2(-50-35s)-1]^3+dw[ur^2(-50-35s)-1]+e$. Maintenant, en utilisant $b = sw^2, d = \frac{sw^4}{2}, e = \frac{s-10}{10}w^5$, nous en déduisons que $v^5+bv^3+dv+e \neq 0 \Leftrightarrow [ur^2(-50-35s)-1]^5+s[ur^2(-50-35s)-1]^3+\frac{s}{2}ur^2(-50-35s)-\frac{2s+5}{5} \neq 0$ ce qui est vrai par définition de $\mathcal{R}_1$. Donc $\varepsilon = \chi(v^5+bv^3+dv+e) \neq 0$
2. x est bien défini puisque $v+w=0 \Leftrightarrow r=0$ et $0 \notin \mathcal{R}_1$
3. Montrons à présent que $F_1(x)$ est un carré non nul. Pour cela, nous allons considérer le cas où $\varepsilon = 1$ ou $\varepsilon = -1$.
   — Si $\varepsilon = 1$ i.e $F_1(v)$ est un carré non nul et $x = v$ alors $F_1(x)$ est un carré non nul.
   — Si $\varepsilon = -1$, on a $x = \left(\frac{w(-v+w)}{v+w}\right)$ et

   $$F_1(x) = \frac{\omega^5(-v+\omega)^5+b\omega^3(-v+\omega)^3(v+\omega)^2+d\omega(-v+\omega)(v+\omega)^4+e(v+\omega)^5}{(v+\omega)^5}.$$

   Comme $b = sw^2, d = \frac{sw^4}{2}$ et $e = \frac{s-10}{10}w^5$ dans $F_1(x)$, après quelques calculs, on obtient :

   $$F_1(x) = \frac{v^5(-7\omega^5\frac{s}{5}-2\omega^5)+v^3(2\omega^7 s-20\omega^7)+v(\omega^9 s-10\omega^9)+8\omega^{10}\frac{s}{5}}{(v+\omega)^5}.$$

   Posons $\alpha_5 = (-7\omega^5\frac{s}{5}-2\omega^5)$, $\alpha_3 = (2\omega^7 s-20\omega^7)$, $\alpha_1 = (\omega^9 s-10\omega^9)$ et $\alpha_0 = 8\omega^{10}\frac{s}{5}$ alors $F_1(x) = \frac{\alpha_5}{(v+\omega)^5}\left[v^5+\frac{\alpha_3}{\alpha_5}v^3+\frac{\alpha_1}{\alpha_5}v+\frac{\alpha_0}{\alpha_5}\right]$.
   On a : $\frac{\alpha_3}{\alpha_5} = \frac{2\omega^7 s-20\omega^7}{-7\omega^5\frac{s}{5}-2\omega^5} = \frac{\omega^7(2s-20)}{\frac{\omega^5}{5}(-7s-10)} = \frac{\omega^2(10s-100)}{-7s-10}$.

   Puisque $7s^2+20s-100=0 \Rightarrow 10s-100 = -7s^2-10$, donc $\frac{\alpha_3}{\alpha_5} = \frac{w^2(-7s^2-10s}{-7s-10} = sw^2 = b$.

   $\frac{\alpha_1}{\alpha_5} = \frac{\omega^9 s-10\omega^9}{-7\omega^5\frac{s}{5}-2\omega^5} = \frac{\omega^9(5s-100)}{\omega^5(-7s-10)} = \frac{\omega^4\frac{10s-100}{2}}{-7s-10} = \frac{\omega^4\frac{-7s^2-10s}{2}}{-7s-10} = s\frac{\omega^4}{2} = d$.

   $\frac{\alpha_0}{\alpha_5} = \frac{8\omega^{10}\frac{s}{5}}{-7\omega^5\frac{s}{5}-2\omega^5} = \frac{8\omega^{10}s}{\omega^5(-7s-10)} = \frac{8s\omega^5}{-7s-10}$. On sait que $7s^2+20s-100=0 \Longrightarrow -7s^2-20s+100=0 \Longrightarrow -7s^2-80s+60s+100=0 \Longrightarrow -7s^2+60s+100=80s \Longrightarrow (-7s-10)(s-10)=80s=10\times 8s \Longrightarrow \frac{s-10}{10} = \frac{8s}{-7s-10}$. Donc on a $\frac{\alpha_0}{\alpha_5} = e$.
   Ainsi, on a $\chi(F_1(x)) = \chi(\alpha_5)\chi(v+\omega))\chi(F_1(v)) = -\chi(\alpha_5(v+w))$ car $F_1(v)$ est non nul et n'est pas un carré. Mais on sait que $\alpha_5(v+w) = \omega^5\frac{-10-7s}{5}(wur^2(-50-35s)) = uw^6r^2(-10-7s)^2 = u\,(rw^3(-10-7s))^2$. Comme $r \neq 0$ dans $\mathcal{R}_1$, alors $\chi(\alpha_5(v+w)) = \chi(u) = -1$. Donc $\chi(F_1(x)) = 1$, et par conséquent $F_1(x)$ est un carré non nul, on en déduit donc que $y = -\varepsilon\sqrt{F_1(x)}$ est bien défini.

□

Lemme 2.3.11
Dans la situation du théorème 2.3.10 et de la définition 2.3.9, on a $\phi_1(r) = \phi_1(-r), \quad \forall r \in \mathcal{R}_1$ *et* $\#(\phi_1^{-1}(\phi_1((r))) = 2, \quad \forall r \in \mathcal{R}_1$.

Preuve :

— Nous avons $\phi_1(r) = (x, y)$ avec $x := \frac{1+\varepsilon}{2}v + \frac{1-\varepsilon}{2}\left(\frac{w(-v+w)}{v+w}\right)$ et $v := w[ur^2(-50-35s)-1]$. Puisque $v = v(r^2)$ alors $\phi_1(r) = \phi_1(-r)$.

— Soient $r, r' \in \mathcal{R}_1$ tels que $\phi_1(r) = \phi_1(r')$. Comme dans l'algorithme 4 pour r (resp r'), on définit v, ε, x, y (resp. v', ε', x', y'). Puisque $x = x'$ et $y = y'$ alors $-\varepsilon\sqrt{F_1(x)} = -\varepsilon'\sqrt{F_1(x)}$. D'où $\varepsilon = \varepsilon'$. De l'égalité $x = x'$, nous en déduisons que $v = v'$ (pour $\varepsilon = \varepsilon' = 1$ et $\varepsilon = \varepsilon' = -1$) donc $r^2 = r'^2$. Ce qui implique que $r' = \pm r$.

□

Algorithme 5 : Hashing-2-Genus2-Inverting

Entrées : La courbe hyperelliptique $\mathbb{H}^1$ et $(x, y) \in \mathbb{H}^1$.
Sortie : $\bar{r}$ tel que $\phi_1(\bar{r}) = (x, y)$, $\bar{r} \in \mathcal{R}_1$ ou $\perp$ (ce qui signifie que (x, y) n'est pas dans $\phi_1(\mathcal{R}_1)$).

Si *$uw(x+w)(-50-35s)$ est un carré non nul* **Alors**
  Si $y = \sqrt{F_1(x)}$ **Alors**
    $\bar{r} := \sqrt{\frac{2w}{u(x+w)(-50-35s)}}$;
  Sinon
    Si $y = -\sqrt{F_1(x)}$ **Alors**
      $\bar{r} := \sqrt{\frac{x+w}{uw(-50-35s)}}$;
    Fin Si
  Fin Si
  Retourner $\bar{r}$;
Sinon
  Retourner $\perp$.
Fin Si

Théorème 2.3.12
Dans la situation du théorème 2.3.10 et de la définition 2.3.9, l'algorithme 5 définit la fonction de décodage de ϕ_1 *(c'est à dire l'algorithme qui permet de déterminer les antécédents des éléments de* $\mathrm{Im}(\phi_1)$*). De plus,* $\mathrm{Im}(\phi_1)$ *est l'ensemble des points* $(x, y) \in \mathbb{H}^1$ *vérifiant* $\chi(uw(x+w)(-50-35s)) = 1$.

Preuve :

1. — Supposons que $(x, y) \in \mathrm{Im}(\phi_1)$.
   — Si $\varepsilon = 1$ alors $x = v$. Donc $x + w = 0 \Leftrightarrow v + w = 0 \Leftrightarrow r = 0$ mais on sait que $0 \notin \mathcal{R}_1$.
   Maintenant, on a $uw(x+w)(-50-35s) = uw(v+w)(-50-35s) = uw[ur^2w(-50-35s)](-50-35s) = u^2w^2r^4(-50-35s)^2$.

— Si $\varepsilon = -1$ alors $x = \frac{w(-v+w)}{v+w} =$, Donc $x + w = 0 \Leftrightarrow \frac{-vw+w^2}{v+w} = -w \Leftrightarrow w = 0$ or $w \in \mathbb{F}_q^*$.
Nous avons donc $uw(x+w)(-50-35s) = uw[\frac{w(-v+w)}{v+w} + w](-50-35s) = \frac{uw}{v+w}(2w^2)(-50-35s) = \frac{2w^2}{r^2}$, puisque $v + w = uwr^2(-50-35s)$ d'après le théorème 2.3.10, 2 est un carré quand $q \equiv 1 \bmod 8$ ou $q \equiv 7 \bmod 8$.
Nous concluons donc que $uw(x+w)(-50-35s)$ est un carré non nul.

— Réciproquement, supposons que $uw(x+w)(-50-35s)$ est un carré non nul dans $\mathbb{F}_q$. Montrons que $(x, y) \in \mathrm{Im}(\phi_1)$. On pose $\bar{r} = \sqrt{\frac{x+w}{uw(-50-35s)}}$ si $y \notin \sqrt{\mathbb{F}_q^2}$ et $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-50-35s)}}$ si $y \in \sqrt{\mathbb{F}_q^2}$. Selon les hypothèses ci-dessus $\frac{x+w}{uw(-50-35s)}$ et $\frac{2w}{u(x+w)(-50-35s)}$ sont bien définis et sont des carrés non nuls, et par conséquent $\bar{r}$ est toujours bien défini.
Maintenant, nous allons prouver que $\bar{r} \in \mathcal{R}_1$ et $(x, y) \in \mathrm{Im}(\phi_1)$. On définit $\bar{v}, \bar{\varepsilon}, \bar{x}, \bar{y}$ comme dans l'algorithme 4.
Si $y \notin \sqrt{\mathbb{F}_q^2}$ alors $\bar{r} = \sqrt{\frac{x+w}{uw(-50-35s)}} \Longrightarrow \bar{v} = w[u\bar{r}^2(-50-35s) - 1] = w[\frac{x+w}{w} - 1] = x$. Donc on a, $\bar{\varepsilon} = \chi(\bar{v}^5 + b\bar{v}^3 + d\bar{v} + e) = \chi(x^5 + bx^3 + dx + e) = 1$. Ainsi $\bar{x} = \frac{1+\bar{\varepsilon}}{2}\bar{v} + \frac{1-\bar{\varepsilon}}{2}\left(\frac{\omega(-\bar{v}+\omega)}{\bar{v}+\omega}\right) = \bar{v} = x$ et $\bar{y} = -\bar{\varepsilon}\sqrt{x^5 + bx^3 + dx + e} = -\sqrt{x^5 + bx^3 + dx + e} = y$.
Maintenant, si $y \in \sqrt{\mathbb{F}_q^2}$ alors $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-50-35s)}}$, puisque $\bar{v} = w[u\bar{r}^2(-50-35s) - 1]$ donc $\bar{v} = w\frac{w-x}{x+w}$.
Après quelques calculs, nous avons :
$\bar{\varepsilon} = \chi(\bar{v}^5 + b\bar{v}^3 + d\bar{v} + e) = \chi(\left(\frac{-7\omega^5\frac{s}{5} - 2\omega^5}{(x+\omega)^5}\right)(x^5 + bx^3 + dx + e)) = \chi(5w(x+w)(-7s-10) = -1$ car $uw(x+w)(-50-35s)$ est un carré non nul et $\chi(u) = -1$.
Comme $\bar{\varepsilon} = -1$ alors $\bar{x} = \frac{\omega(-\bar{v}+\omega)}{\bar{v}+\omega} = x$ et $\bar{y} = -\bar{\varepsilon}\sqrt{x^5 + bx^3 + dx + e} = \sqrt{x^5 + bx^3 + dx + e} = y$. Dans les deux cas, nous avons $\bar{x} = x$ et $\bar{y} = y$.
Comme pour tout v, on a $v^5 + bv^3 + dv + e \neq 0$, donc $[u\bar{r}^2(-50-35s) - 1]^5 + s[u\bar{r}^2(-50-35s) - 1]^3 + \frac{s}{2}[u\bar{r}^2(-50-35s) - 1] \neq \frac{2s+5}{5}$, d'où $\bar{r} \in \mathcal{R}_1$.

2. Découle de la preuve précédente.

□

Remarque 2.3.13 (Extension sur $\mathbb{F}_q$)
Par construction, ϕ_1 est défini sur $\mathcal{R}_1 = \mathbb{F}_q \backslash \mathcal{S}_1$.

— *On se propose de prolonger ϕ_1 en 0 de la manière suivante. Rappelons que $b = sw^2$, $d = \frac{sw^4}{2}$, $e = \frac{s-10}{10}w^5$. Donc $F_1(-w) = -w^5 - sw^5 - \frac{sw^5}{4} + \frac{s-10}{10}w^5 = (-1 - s - \frac{s}{2} + \frac{s-10}{10})w^5 = \frac{-14s-11}{10}w^5$. Maintenant, on choisit w de telle sorte que $F_1(-w)$ soit un carré non nul et on pose $\phi_1(0) = (-w, \sqrt{F_1(-w)})$.*

— *On a $v(r) = w[ur^2(-50-35s) - 1]$. On pose $Z_1 = \{r \in \mathbb{F}_q^* : F_1(v(r)) = 0\}$. Si Z_1 n'est pas vide, on pose $\phi_1(\pm r) = (v(r), 0)$ pour tout $r \in Z_1$ pour conserver la presque injectivité (on pouvait aussi poser $\phi_1(\pm r) = (-w, -\sqrt{F_1(-w)})$ $\forall r \in Z_1$ avec le choix ci-dessus de w).*

.

2.3.2 Encodage presque-injectif sur $\mathbb{H}^2$

Soit $q = 5^n$ une puissance de 5 et $u \in \mathbb{F}_q^*$ un carré non nul. Soient $w \in \mathbb{F}_q^*$ un paramètre arbitraire et $\mathbb{H}^2$ une courbe hyperelliptique de genre 2 donnée par

$$\mathbb{H}^2 : y^2 = F_2(x) = x^5 + ax^4 + e$$

sur $\mathbb{F}_q$ où $a \in \mathbb{F}_q^*$ et $e = 4aw^4$.
Soit u un paramètre tel que $\chi(u) = -1$ et on définit l'ensemble
$\mathcal{R}_2 = \left\{r \in \mathbb{F}_q^*, w(1-ur^2)^5 + a(1-ur^2)^4 + 4a \neq 0\right\}$.

Algorithme 6 : Hashing-3-Genus2-Encoding

Entrées : La courbe hyperelliptique $\mathbb{H}^2$, un élément $r \in \mathcal{R}_2$.
Sortie : Un point (x, y) de $\mathbb{H}^2$

1 $v = v(r^2) := w(1-ur^2)$;
2 $\varepsilon := \chi(v^5 + av^4 + 4aw^4)$;
3 $x := \dfrac{1+\varepsilon}{2}v + \dfrac{1-\varepsilon}{2}\left(\dfrac{-wv}{v-w}\right)$;
4 $y := -\varepsilon\sqrt{x^5 + ax^4 + 4aw^4}$;
5 **Retourner** (x, y).

Définition 2.3.14.

Dans la situation de l'algorithme 6, l'encodage pour la courbe hyperelliptique $\mathbb{H}^2$ est la fonction $\phi_2 : \mathcal{R}_2 \to \mathbb{H}^2 : r \mapsto \phi_2(r) = (x, y)$.

Théorème 2.3.15
L'algorithme 6 détermine un encodage presque-injectif $\phi_2 : \mathcal{R}_2 \to \mathbb{H}$ où $\mathcal{S}_2 = \mathbb{F}_q \backslash \mathcal{R}_2$ est un sous-ensemble de taille au maximum de 11 éléments.

Preuve :

- x est bien défini puisque $v - w = 0 \Leftrightarrow v = w \Leftrightarrow w - wur^2 = w \Rightarrow w = 0$ ou $r = 0$, ce qui est impossible.
- Puisque $v = w(1-ur^2)$, par définition de $\mathbb{R}_2$, on a $v^5 + av^4 + 4aw^4 = w^4\left[w(1-ur^2)^5 + a(1-ur^2) + 4a\right] \neq 0$. Donc $\varepsilon \neq 0$.
- Maintenant, montrons que $F_2(x)$ est un carré non nul.
  - Si $\varepsilon = 1$, on a $x = v$ et $\chi(F_2(v)) = 1$. Par conséquent $F_2(v)$ est un carré non nul.
  - Si $\varepsilon = -1$, alors $F_2(v)$ est non nul et n'est pas un carré, comme $x = \dfrac{-wv}{v-w}$, donc en le remplaçant dans $x^5 + ax^4 + 4aw^4$, on obtient après quelques calculs $\chi(F_2(x)) = \chi(\dfrac{-w^5}{(v-w)^5})\chi(F_2(v))$. Donc $\chi(F_2(v)) = -1$, et $\chi(F_2(x)) = -\chi(\dfrac{-w}{v-w}) = -\chi(-w\,(w - uwr^2 - w)) = -\chi(uw^2r^2) = -\chi(u) = 1$. Ainsi, $F_2(x)$ est un carré non nul.

Donc dans tous les cas, $F_2(x)$ est un carré non nul.

□

Lemme 2.3.16
Dans la situation du théorème 2.3.15 et de la définition 2.3.14, on a $\phi_2(r) = \phi_2(-r), \quad \forall r \in \mathcal{R}_2$ *et* $\#(\phi_2^{-1}(\phi_2(r))) = 2, \quad \forall r \in \mathcal{R}_2$.

Preuve : Similaire à la preuve du lemme 2.3.12.

□

Algorithme 7 : Hashing-3-Genus2-Inverting

Entrées : La courbe hyperelliptique $\mathbb{H}^2$ et $(x, y) \in \mathbb{H}^2$.
Sortie : $\bar{r}$ tel que $\phi_2(\bar{r}) = (x, y)$, $\bar{r} \in \mathcal{R}_2$ ou $\perp$ (ce qui signifie que (x, y) n'est pas dans $\phi_2(\mathcal{R}_2)$.

Si $y = \sqrt{F_2(x)}$ **Alors**
  Si $uw(w + x)$ *est un carré non nul* **Alors**
    $\bar{r} = \sqrt{\dfrac{w}{u(w+x)}}$;
    Retourner $\bar{r}$;
  Sinon
    Retourner $\perp$
  Fin Si
Sinon
  Si $uw(w - x)$ *est un carré non nul* **Alors**
    $\bar{r} = \sqrt{\dfrac{w-x}{uw}}$;
    Retourner $\bar{r}$;
  Sinon
    Retourner $\perp$
  Fin Si
Fin Si

Théorème 2.3.17
Dans la situation du théorème 2.3.15 et de la définition 2.3.14, l'algorithme 7 définit la fonction de décodage de ϕ_2. *De plus,* $\mathrm{Im}(\phi_2)$ *est l'ensemble des points* $(x, y) \in \mathbb{H}^2$ *vérifiant* $\chi(uw(w - x)) = 1$ *si* $y \notin \sqrt{\mathbb{F}_q^2}$ *et* $\chi(uw(w + x)) = 1$ $y \in \sqrt{\mathbb{F}_q^2}$.

Preuve :

1. (a) Supposons que $(x, y) \in \mathrm{Im}(\phi_2)$, il existe $\bar{r}$ tel que $\phi_2(\bar{r}) = (x, y)$ d'après le théorème 2.3.15.
   — $y \notin \sqrt{\mathbb{F}_q} \Rightarrow \varepsilon = 1$ donc $x = v$. Puisque $v = w(1 - ur^2)$ et $r \neq 0$, on en déduit que $(w - x)uw = (w - v)uw = wur^2uw = uw^2r^2$ est un carré non nul.
   — $y \in \sqrt{\mathbb{F}_q} \Rightarrow \varepsilon = -1$ alors $x = \dfrac{-wv}{v - w}$. Comme $v = w(1 - ur^2)$ et $r \neq 0$, alors $uw(w + x) == \dfrac{1}{r^2}$ est un carré non nul.

(b) Réciproquement, supposons que $uw(w-x)$ est un carré et $w-x \neq 0$ si $y \notin \sqrt{\mathbb{F}_q}$, et $uw(w+x)$ est un carré et $w+x \neq 0$ si $y \in \sqrt{\mathbb{F}_q}$. Maintenant, notre but est de montrer que $\bar{r}$ est défini et $\phi_4(\bar{r}) = (x, y)$. . On définit $\bar{v}, \bar{\varepsilon}, \bar{x}, \bar{y}$ comme dans l'algorithme 6. Ils y'a deux cas à considérer :

— Premier cas : $y \notin \sqrt{\mathbb{F}_q^2}$. Pour ce cas, nous avons $\bar{r} = \sqrt{\dfrac{w-x}{uw}}$. Alors $\bar{r}$ est bien défini puisque $uw(w-x)$ est un carré et $\bar{r} \neq 0$ puisque $w - x \neq 0$ par hypothèse. Donc $\bar{v} = w - uw\bar{r}^2 = w - uw\left(\dfrac{w-x}{uw}\right) = w - w + x = x$; ainsi $\bar{\varepsilon} = \chi(\bar{v}^5 + a\bar{v}^4 + 4aw^4) = \chi(\bar{x}^5 + a\bar{x}^4 + 4aw^4) = 1$. ce qui implique que $\bar{x} = \dfrac{1+\bar{\varepsilon}}{2}\bar{v} + \dfrac{1-\bar{\varepsilon}}{2}\left(\dfrac{-w\bar{v}}{\bar{v}-w}\right) = \bar{v} = x$ et $\bar{y} = -\bar{\varepsilon}\sqrt{F_2(\bar{x})} = -\sqrt{F_2(x)} = y$.

— Deuxième cas : $y \in \sqrt{\mathbb{F}_q^2}$. Alors $\bar{r} = \sqrt{\dfrac{w}{u(w+x)}}$ est bien défini puisque $uw(w+x)$ est un carré et $\bar{r} \neq 0$ puisque $w(w+x) \neq 0$ par hypothèse, donc $\bar{v} = w - uw\bar{r}^2) = w - uw\left(\dfrac{w}{u(x+w)}\right) = \dfrac{wx}{w+x}$.

On a $\bar{v}^5 + a\bar{v}^4 + 4aw^4 = \dfrac{(wx)^5 + a(wx)^4(w+x) + 4aw^4(w+x)^5}{(w+x)^5}$. Donc $\bar{\varepsilon} = \chi(\bar{v}^5+a\bar{v}^4+4aw^4) = \chi(\dfrac{w^5}{(x+w)^5}(x^5+ax^4+4aw^4)) = \chi(\dfrac{w^5}{(x+w)^5})\chi(F_2(x)) = \chi(w(w+x)) = \chi(u) = -1$ puisque $uw(w+x)$ et $F_2(x)$ sont des carrés non nuls. Donc nous avons $\bar{x} = \dfrac{1+\bar{\varepsilon}}{2}\bar{v} + \dfrac{1-\bar{\varepsilon}}{2}\left(\dfrac{-w\bar{v}}{\bar{v}-w}\right) = \dfrac{-w\bar{v}}{\bar{v}-w} = \dfrac{\dfrac{-w^2x}{w+x}}{\dfrac{wx}{w+x} - w} = \dfrac{-w^2x}{-w^2} = x$ et $\bar{y} = -\bar{\varepsilon}\sqrt{F_2(\bar{x})} = \sqrt{F_2(x)} = y$.

Comme pour tout v, nous avons $v^5 + av^4 + e \neq 0$, donc $w(1-ur^2)^5 + a(1-ur^2)^4 + 4a \neq 0$, ainsi $\bar{r} \in \mathcal{R}_2$.

2. Découle de la preuve ci-dessus.

□

Remarque 2.3.18 (Extension sur $\mathbb{F}_{5^n}$)

— *Comme pour ϕ_1, nous proposons d'étendre ϕ_2 en 0 comme suit. On sait que $e = 4aw^4$ et $F_2(w) = w^5$. En choisissant maintenant $w = z^2$ alors $F_2(w)$ est un carré non nul. On pose $\phi_2(0) = (w, \sqrt{F_2(w)})$.*

— *On a $v(r) = w(1-ur^2)$. On pose $Z_2 = \{r \in \mathbb{F}_q^* : F_2(v(r)) = 0\}$. Si Z_2 n'est pas vide, on pose $\phi_2(\pm r) = (v(r), 0)$ pour tout $r \in Z_2$ pour conserver la presque injectivité (on pouvait aussi poser $\phi_2(\pm r) = (w, -\sqrt{F_2(w)})$ $\forall r \in Z_2$ avec le choix ci-dessus de w).*

2.4 Applications : Hachage indifférentiable sur la Jacobienne

2.4.1 Méthode générale pour hacher sur la Jacobienne

Considérons une fonction d'encodage F sur une courbe C, et $\mathbb{J}\mathrm{ac}$ la Jacobienne de C. Supposons que C possède au moins un point $\mathbb{F}_q-$rationnel $\mathcal{O}$; on peut alors fixer un plongement $C \to \mathbb{J}\mathrm{ac}$, qui envoie un point P sur le diviseur $(P) - (\mathcal{O})$ degré 0. Les propriétés de régularité de la fonction $F^{\otimes s}$ peuvent découler formellement du comportement de F. $F^{\otimes s} : (\mathbb{F}_q)^s \to \mathbb{J}\mathrm{ac}(\mathbb{F}_q), (t_1, \ldots, t_s) \mapsto F(t_1) + \ldots F(t_s)$.

Farashahi et *al.* [FFS+13] ont ensuite montré avec leur technique que les constructions de fonctions de hachage de la forme générale :

$$H(m) = F(h_1(m)) + \ldots F(h_s(m))$$

se comportent bien (c'est à dire qu'elles vérifient l'indifférentiabilité), et ce, dès que F est un encodage bien distribué, s un entier supérieur au genre de la courbe (soit $s \geqslant 3$ pour le genre 2) et les $h_i, i = 1, \ldots, s$ sont des fonctions de hachage classiques modélisées comme des oracles aléatoires.

Soit χ_q un caractère du groupe abélien $\mathbb{J}\mathrm{ac}(\mathbb{F}_q)$. Nous définissons les sommes de caractères

$$S_F(\chi_q) = \sum_{t \in \mathbb{F}_q} \chi_q(F(t)). \tag{2.2}$$

Lemme 2.4.19 (distance statistique)
Si $F : \mathbb{F}_q \to C(\mathbb{F}_q)$ est un encodage B-bien-distribué sur une courbe C, alors pour tout $D \in \mathbb{J}\mathrm{ac}(\mathbb{F}_q)$, la distance statistique entre la distribution définie par la construction $F^{\otimes s}$ et la distribution uniforme sur $\mathbb{J}\mathrm{ac}(\mathbb{F}_q)$ est bornée comme suit :

$$\sum_{D \in \mathbb{J}\mathrm{ac}(\mathbb{F}_q)} \left| \frac{N_s(D)}{q^s} - \frac{1}{\#\mathbb{J}\mathrm{ac}(\mathbb{F}_q)} \right| \leqslant \frac{B^s}{q^{s/2}} \sqrt{\#\mathbb{J}\mathrm{ac}(\mathbb{F}_q)}$$

où $N_s(D)$ représente le nombre d'antécédents de D par $F^{\otimes s}$: $N_s(D) = \#\{(t_1, \ldots, t_s) \in (\mathbb{F}_q)^s | D = F(t_1) + \ldots F(t_s)\}$.

Preuve : Voir [FFS+13]

□

Nous allons maintenant utiliser ϕ_1 pour construire des fonctions de hachage indifférentiables sur $\mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)$ la Jacobienne de $\mathbb{H}_1$. Nous allons appliquer la technique de Farashahi et *al.* [FFS+13].

2.4.2 Hachage indifférentiable sur la Jacobienne de $\mathbb{H}_1$

Considérons l'encodage ϕ_1 sur la courbe hyperelliptique $\mathbb{H}^1 : y^2 = F_1(x) = x^5 + bx^3 + dx + e$ avec $b = sw^2$, $d = \frac{sw^4}{2}$, $e = \frac{s-10}{10} w^5$ où $w, s \in \mathbb{F}_q^*$ tels que $7s^2 + 20s - 100 = 0$ lorsque $q = 7 \bmod 8$, comme défini dans la sous-section 2.3.1.

D'après le théorème 2.3.17, nous avons ce qui suit :

$$y \notin \sqrt{\mathbb{F}_q^{*2}} \Longleftrightarrow r^2 := \frac{x+w}{uw(-50-35s)} \qquad (1*)$$

$$y \in \sqrt{\mathbb{F}_q^{*2}} \Longleftrightarrow r^2 := \frac{2w}{u(x+w)(-50-35s)} \qquad (2*)$$

D'après l'algorithme 4, la fonction d'encodage pour la courbe hyperelliptique $\mathbb{H}^1$ est la fonction $\phi_1 : \mathcal{R}_1 \to \mathbb{H}^1 : r \mapsto \phi_1(r) = (x, y)$ et vérifie $\phi_1(r) = \phi_1(-r)$ où

$$\mathcal{R}_1 = \left\{ r \in \mathbb{F}_q^*, [ur^2(-50-35s)-1]^5 + s[ur^2(-50-35s)-1]^3 + \frac{s}{2}ur^2(-50-35s) \neq \frac{2s+5}{5} \right\}$$

Nous allons utiliser ces résultats pour montrer que notre fonction d'encodage ϕ_1 est $B-$*bien* distribuée. La preuve sera similaire à celle de Farashahi et *et al.* ([FFS+13] sous-section 5.3).

Théorème 2.4.20
Supposons que $q = 3 \bmod 4$. *Soit* ϕ_1 *la fonction d'encodage décrite ci-dessus. Pour tout caractère non trivial* χ *de* $\mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)$, *la somme de caractères* $S(\chi)$ *donnée par (2.2) satisfait*

$$|S_{\phi_1}(\chi)| \leqslant 32\sqrt{q} + 119$$

Preuve :
Posons $\mathcal{S}_1 = \mathbb{F}_q \backslash \mathcal{R}_1$, $f_0(x,r) = r^2 - \frac{x+w}{uw(-50-35s)}$ et $f_1(x,r) = r^2 - \frac{2w}{u(x+w)(-50-35s)}$. Introduisons les courbes projectives lisses

$$C_{1,j} = \left\{ (x,y,r) : (x,y) \in \mathbb{H}^1, f_j(x,r) = 0 \right\} \text{ pour } j = 0,1 \qquad (3*)$$

Puisque $q \equiv 7 (\bmod 8)$ alors $y \in \sqrt{\mathbb{F}_q^{*2}} \Leftrightarrow \chi_q(y) = 1$ et $y \notin \sqrt{\mathbb{F}_q^{*2}}, y \neq 0 \Leftrightarrow \chi_q(y) = -1$. Par les équations (1*), (2*) et (3*), nous définissons les recouvrements suivants $h_{1,j} : C_{1,j} \to \mathbb{H}_1$, $j = 0, 1$. Comme dans [FFS+13], la fonction rationnelle r sur $C_{1,j}$ permet de définir un morphisme $g_{1,j} : C_{1,j} \to \mathbb{P}^1$ tel que tout point dans $\mathbb{A}^1 \backslash \mathcal{S}_1$ possède exactement deux antécédents (qui sont conjuguées via $y \to -y$) dans une des deux courbes $C_{1,j}, j = 0, 1$.

Puisque $q \equiv 3 (\bmod 4)$, alors y ou $-y$ est un carré donc un seul des antécédents précédents vérifie $\chi(y) = (-1)^j$ sur $C_{1,j}$ pour l'un des $j = 0, 1$ et aucun antécédent pour l'autre j. Soit $P \in C_j(\mathbb{F}_q)$ cet antécédent.

Donc, $\phi_1(r) = h_{1,j}(P)$. Montrons que ϕ_1 est un encodage bien distribué. On note par $\mathbb{J}\mathrm{ac}_1$ la Jacobienne de $\mathbb{H}_1$, et on fixe un plongement $\mathbb{H}_1 \to \mathbb{J}\mathrm{ac}_1$. On peut supposer que S_1 est un sous-ensemble de $\mathbb{P}^1$ et on pose $S_{1j} = g_{1,j}^{-1}(S_1 \cup \infty)$.

Pour tout $r \in \mathcal{R}_1$, la somme des caractères $S_{\phi_1}(\chi)$ peut être écrite comme :

$$\sum_{r \in \mathcal{R}_1} \chi(\phi_1(r)) = \sum_{P \in C_{10}(\mathbb{F}_q) \backslash S_{10}, \chi_q(y) = +1} \chi(h_{10}(P)) + \sum_{P \in C_{11}(\mathbb{F}_q) \backslash S_{11}, \chi_q(y) = -1} \chi(h_{11}(P))$$

On remarque que

$$\sum_{P\in C_{1,j}(\mathbb{F}_q)} \chi(h_{1,j}(P)).\left(\frac{1+(-1)^j\chi_q(y)}{2}\right) = \sum_{P\in C_{1,j}(\mathbb{F}_q)\chi_q(y)=(-1)^j} \chi(h_{1,j}(P)) + \frac{1}{2}\sum_{P\in C_{1,j}(\mathbb{F}_q)\chi_q(y)=0} \chi(h_{1,j}(P))$$

L'expression $\frac{1}{2}\sum_{P\in C_{1,j}(\mathbb{F}_q)\chi_q(y)=0} \chi(h_{1,j}(P))$ contient au plus $2\times 5 = 10$ termes.

Déterminons $\sum_{P\in C_{1,j}(\mathbb{F}_q)\chi_q(y)=(-1)^j} \chi(h_{1,j}(P)) = \sum_{P\in C_{1,j}(\mathbb{F}_q)} \chi(h_{1,j}(P)).\left(\frac{1+(-1)^j\chi_q(y)}{2}\right) - \frac{1}{2}\sum_{P\in C_{1,j}(\mathbb{F}_q)\chi_q(y)=0} \chi(h_{1,j}(P))$

Puisque h_0 et h_1 sont totalement ramifiés sur les points dans $\mathbb{H}_1$ tels que $x=-w$, ils ne peuvent donc pas se factoriser à travers un recouvrement qui n'est pas ramifié de $\mathbb{H}_1$. Donc d'après le Théorème (2.1),

$$\left|\sum_{P\in C_j(\mathbb{F}_q)} \chi(h_j(P)).\left(\frac{1+(-1)^j\chi(y)}{2}\right)\right| \leqslant (2g_{C_j}-2+\deg(y))\sqrt{q}$$

Déterminons le genre g_{C_j} de C_j.

$h_{1,j}: C_j \to \mathbb{H}_1$ est seulement ramifié en $x=-w$. Par conséquent, d'après la formule de Riemann-Hurwitz, nous obtenons $2g_{C_j}-2 = 2(2(2)-2)+2.(2-1) = 6$, ainsi les courbes C_j sont de genre 4.

Maintenant, nous allons déterminer le degré $\deg(y)$ de y comme une fonction rationnelle sur C_j. Nous avons :

$$\deg(y) = [\mathbb{F}_q(x,y,r):\mathbb{F}_q(x,y)].[\mathbb{F}_q(x,y):\mathbb{F}_q(y)] = 2.5 = 10$$

Ainsi :

$$\left|\sum_{P\in C_j(\mathbb{F}_q)} \chi(h_j(P)).\left(\frac{1+(-1)^j\chi(y)}{2}\right)\right| \leqslant 16\sqrt{q}$$

Par conséquent, en utilisant les formules précédentes, nous avons

$$\left|\sum_{r\in\mathcal{R}_1}\chi(\phi_1(r))\right| \leqslant (16\sqrt{q}+\#S_{10}+5)+(16\sqrt{q}+\#S_{11}+5) = (32\sqrt{q}+10+\#S_{10}+\#S_{11})$$

Donc

$$|S_{\phi_1}(\chi)| \leqslant 32\sqrt{q}+10+\#S_{10}+\#S_{11}+\#S_1$$

On a $\#S_1 = 1+2.10 = 21$. Puisque $g_{1,j}$ est une application de degré 2, $\#S_{1,j} \leqslant 2(\#S+1) \leqslant 44$.

Donc

$$|S_{\phi_1}(\chi)| \leqslant 32\sqrt{q}+10+44+44+21 = 32\sqrt{q}+119$$

comme souhaité. En d'autres termes, ϕ_1 est un encodage $(32+119q^{-1/2})$-bien-distribué. □

Théorème 2.4.21
Supposons que $q = 3 \bmod 4$. *Soit* $\mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)$ *la jacobienne de* H_1. *Pour tout* $D \in \mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)$, *on note par* $N_3(D)$ *le nombre d'antécédents de* D *par* $\phi_1^{\otimes 3}$.

La distance statistique entre la distribution définie par $\phi_1^{\otimes 3}$ *et la distribution uniforme dans* $\mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)$ *est bornée comme suit :*

$$\sum_{D \in \mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)} \left| \frac{N_3(D)}{q^3} - \frac{1}{\#\mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)} \right| \leqslant \frac{(32 + 119q^{-1/2})^3}{q^{3/2}} \sqrt{\#\mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)}.$$

Preuve : Elle découle du théorème ci-dessus 2.4.20 et de la preuve dans [FFS+13]. □

Par conséquent, la distribution définie par $\phi_1^{\otimes 3}$ sur $\mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)$ est statistiquement indistinguable de la distribution uniforme. En particulier, la construction suivante :

$$m \mapsto H(m) = \phi_1(h_1(m)) + \phi_1(h_2(m)) + \phi_1(h_3(m)) \in \mathbb{J}\mathrm{ac}_1(\mathbb{F}_q)$$

est indifférentiable d'un oracle aléatoire lorsque h_1, h_2, h_3 sont considérés comme des oracles aléatoires dans $\mathbb{F}_q$.

Conclusion

Dans ce chapitre, nous avons construit de nouveaux encodages dans la jacobienne de deux familles de courbes hyperelliptiques. Ces encodages, qui ne sont pas couverts par les travaux précédents dans ce domaine, ont des propriétés intéressantes, comme la *presque*-injectivité. Ceci peut être utilisé pour construire des fonctions de hachage efficaces et indifférentiables dans la Jacobienne des courbes sous-jacentes. Néanmoins, même si l'on peut utiliser nos encodages pour encoder un message dans la Jacobienne, il serait intéressant de décider si ces familles spéciales de courbes hyperelliptiques conviennent ou non aux préoccupations cryptographiques.

Chapitre 3

Formules unifiées pour certains encodages déterministes presque-injectifs sur des courbes hyperelliptiques [1]

Contents

1. Ce chapitre est basé sur l'article publié suivant :

Seck M., Diarra N. : Unified Formulas for Some Deterministic Almost-Injective Encodings into Hyperelliptic Curves. In : Joux A., Nitaj A., Rachidi T. (eds) Progress in Cryptology - AFRICACRYPT 2018. AFRICACRYPT 2018. Lecture Notes in Computer Science, vol 10831. Springer, Cham (2018)

Récemment, des encodages efficaces déterministes et inversibles sur certaines courbes hyperelliptiques en genre 1 et 2 utilisant la technique dans Elligator 2 (ACM CCS 2013) ont été proposés. Nous avons réussi à généraliser leurs encodages pour des courbes hyperelliptiques de genre $3, 4$ et 5. Nous avons proposé des formules unifiées (utilisant les nombres de Mersenne) pour les encodages sur les courbes hyperelliptiques de genre $g \leqslant 5$:

$$\mathbb{H}_g : y^2 = f_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0$$

Nous conjecturons que notre méthode fonctionne sur n'importe quel genre g.

3.1 Introduction

Pour la construction de protocoles ou de schémas cryptographiques [BF01, CC03] en cryptographie basée sur des courbes (hyper)elliptiques, il est parfois nécessaire de pouvoir représenter une chaîne de bits comme un point d'une courbe (hyper)elliptique [Kob89, MhWZ98, SV08]. Dans Elligator 2 (proposé par Bernstein *et al.* : ACM CCS 2013 [BHKL13]), les points de courbes elliptiques uniformes peuvent être représentés par des chaînes de bits aléatoires uniformes. De nombreux auteurs ont étudié le problème de la conception d'encodages déterministes sur des courbes (hyper)elliptiques. Au cours de la dernière décennie, de nombreux encodages déterministes ont été proposés pour certaines familles de courbes elliptiques et hyperelliptiques.

- En 2006, Shallue et van de Woestijne [SvdW06] ont publié le premier type de fonction d'encodage sur les courbes elliptiques ordinaires (courbes données par une équation de la forme E : $y^2 = x^3 + ax + b$). Ulas [Ula07], en 2007, a généralisé et simplifié l'encodage de Shallue et van de Woestijne en encodant sur les courbes

  $$C : y^2 = x^n + ax + b \text{ et } C' : y^2 = x^n + ax^2 + bx$$

  . Ils ont été les premiers à publier un encodage sur une famille de courbes hyperelliptiques de genre quelconque.
- À CRYPTO 2009, Thomas Icart [Ica09] a défini une fonction d'encodage déterministe sur la courbe elliptique $E : y^2 = x^3 + ax + b$ sur un corps fini $\mathbb{F}_q$ tel que $q \equiv 2 \bmod 3$. Kammerer, Lercier et Renault [KLR10] ont introduit en 2010 une série de nouveaux encodages basés sur les mêmes principes que la fonction d'Icart, à savoir la résolution d'équations de courbes en radicaux. Ils ont proposé un encodage sur la courbe Hessienne $E : x^3 + y^3 + 1 = 3dxy, d \neq 1$ sur $\mathbb{F}_q$ et la courbe hyperelliptique $H_{1,a,b} : y^2 = (x^3 + 3ax + 2)^2 + 8bx^3$ sur $\mathbb{F}_q$ avec $q \equiv 2 \bmod 3$.
- En 2013, Pierre Alain Fouque, Antoine Joux et Mehdi Tibouchi [FJT13] ont proposé une nouvelle construction géométrique essentiellement optimale pour une large classe de courbes comme la courbe hyperelliptique $H_d^\lambda : y^2 = \lambda ax^5 + (c^2 + 1/c^2)x^3 + x$ sur $\mathbb{F}_q$. Dans la même année, Daniel J. Bernstein *et al.* [BHKL13] proposa deux encodages à savoir Elligator 1 et Elligator 2. Dans Elligator 1, ils ont introduit un encodage sur la courbe d'Edwards $E : y^2 + x^2 = 1 + dx^2y^2$ sur $\mathbb{F}_q$, et dans Elligator 2, ils ont proposé un encodage sur la courbe $E : y^2 = x^3 + Ax^2 + Bx$ avec $AB(A^2 - 4B) \neq 0$ sur $\mathbb{F}_q$ où $q = p^n, p > 2$.

- En 2015, Xiaoyang He, Wei Yu et Kunpeng Wang [HYW15] ont proposé deux encodages déterministes de $\mathbb{F}_q$ sur des courbes de Huff généralisées. Yu Wei *et al.* [YWL+16], en 2016, ont aussi proposé un encodage déterministe d'un corps fini $\mathbb{F}_q$ sur une courbe d'Edwards tordue E quand $q \equiv 2 \bmod 3$.
- En 2017, à Africacrypt-2017, Seck *et al.* [SBDK17] ont proposé trois fonctions d'encodage sur trois familles différentes de courbes hyperelliptiques en utilisant la technique d'Elligator 2 de Daniel J. Bernstein *et al.* [BHKL13]. Ils ont montré qu'on peut utiliser leurs encodages pour construire des fonctions de hachage déterministes indifférentiables sur la Jacobienne de certaines familles de courbes hyperelliptiques en utilisant le résultat de Farashahi *et al.* [FFS+13].

Contributions : Les principaux résultats de ce chapitre sont basés sur les travaux de Bernstein *et al.* [BHKL13] dans Elligator 2 et de Seck *et al.* [SBDK17].

Notre première contribution est la construction de quatre fonctions d'encodage déterministes presque injectives $\psi_i : \mathcal{R}_i \subset \mathbb{F}_q \to \mathbb{H}_i$ pour $i = 1, 3, 4, 5$, où :

$$\mathbb{H}_1 : y^2 = f_1(x) = x^3 + a_1 x + a_0;$$

$$\mathbb{H}_3 : y^2 = f_3(x) = x^7 + a_5 x^5 + a_3 x^3 + a_1 x + a_0;$$
$$\mathbb{H}_4 : y^2 = f_4(x) = x^9 + a_7 x^7 + a_5 x^5 + a_3 x^3 + a_1 x + a_0;$$
$$\mathbb{H}_5 : y^2 = f_5(x) = x^{11} + a_9 x^9 + a_7 x^7 + a_5 x^5 + a_3 x^3 + a_1 x + a_0;$$

Leur presque-injectivité signifie que tout point dans le co-domaine $\mathbb{H}_g$ a exactement deux pré-images $(r, -r)$ dans le domaine $\mathbb{F}_q$. Notez que dans le cas où q est un premier, on peut restreindre le domaine à $\{0, \ldots, (q-1)/2\}$ afin d'avoir un encodage injectif pour chaque $\mathbb{H}_g$. Nous montrons également que l'on peut inverser ces encodages sous certaines conditions.

Notre deuxième contribution est la construction d'une formule unifiée des encodages ci-dessus. Nous construisons un encodage déterministe presque-injectif $\psi_g : \mathcal{R}_g \subset \mathbb{F}_q \to \mathbb{H}_g$ où g est un entier non nul inférieur ou égal à cinq. Cet encodage généralise l'encodage dans [SBDK17] $\psi_2 : \mathcal{R}_2 \subset \mathbb{F}_q \to \mathbb{H}_2$ où $\mathbb{H}_2 : y^2 = f_2(x) = x^5 + a_3 x^3 + a_1 x + a_0$ et nos quatre fonctions d'encodage $\psi_i, i = 1, 3, 4, 5$.

Notons que tous ces encodages dépendent d'un paramètre s qui satisfait l'équation du second degré $\alpha_g s^2 + \beta_g s - \gamma_g = 0$ où α_g, β_g et γ_g dépendent uniquement du genre g de la courbe hyperelliptique $\mathbb{H}_g$. La motivation principale de la construction de l'encodage ψ_5 est qu'après avoir encodé sur les courbes hyperelliptiques $\mathbb{H}_i$ pour $i = 3, 4$, on a noté que $\alpha_2 = 3$ est premier, $\alpha_3 = 31$ est premier et $\alpha_4 = 127$ est aussi un entier premier. Nous avons d'abord conjecturé que α_g est premier pour tout genre $g \geqslant 2$. Malheureusement, $\alpha_5 = 511$ n'est pas premier. Nous remarquons que $\alpha_g = 2^{2g-1} - 1$ est un nombre de Mersenne pour $g \in \{1, 2, 3, 4, 5\}$.

En se basant sur ces constructions, nous conjecturons que notre méthode peut être étendue pour un genre arbitraire g même si pour une utilisation en cryptographie dans le cas du problème du logarithme discret (PLD), il faut se limiter à $g \leqslant 4$. Donc notre encodage n'a qu'un intérêt théorique pour $g \geqslant 5$. L'importance de nos résultats se situe donc plutôt du côté théorique dans le cas de courbes hyperelliptiques basées sur le PLD.

Organisation du chapitre : Ce chapitre est organisé comme suit.

— Dans la section 3.2 : Nous donnons nos nouveaux encodages sur les courbes hyperelliptiques $\mathbb{H}_i,\ i = 1, 3, 4, 5$. Nous montrons également dans cette section dans quelles conditions nos encodages sont inversibles.
— Dans la section 3.3 : Nous donnons une généralisation de toutes les fonctions d'encodage sur les courbes hyperelliptiques $\mathbb{H}_i,\ i = 1, 2, 3, 4, 5$.
— Et dans la section 3.4 : nous étendons notre résultat à $g \leqslant 9$. Et nous concluons dans la section 3.5.
— Dans l'annexe 3.6, nous donnons une implémentation de notre encodage unifié en utilisant le logiciel de calcul scientifique SageMath [Dev17].

3.2 Nouveaux encodages presque-injectifs et inversibles sur des courbes hyperelliptiques

Dans cette section, nous proposons quatre fonctions d'encodage presque-injectives sur quatre familles de courbes hyperelliptiques $\mathbb{H}_3, \mathbb{H}_4, \mathbb{H}_5$ et $\mathbb{H}_1$ en utilisant l'approche de Bernstein *et al.* [BHKL13] dans l'Elligator 2 pour les courbes elliptiques. Nous utilisons la même approche que Seck *et al.* ([SBDK17] **Section 3**) afin de trouver des formules unifiées de nos encodages et ceux de Seck *et al.*. Comme mentionné dans [SBDK17], nos nouveaux encodages ont la même complexité asymptotique, à savoir $\mathcal{O}(\log^{2+\circ(1)} q)$.
Nous supposons, dans cette section, que
— $\mathbb{F}_q$ est un corps fini avec $\mathrm{char}(\mathbb{F}_q) = p$ et $q = p^n$ est une puissance d'un nombre premier impair. On suppose que $q \equiv 7 \bmod 8$, alors 2 est un carré ;
— $w \in \mathbb{F}_q^\star$ est un paramètre arbitraire ;
— $u \in \mathbb{F}_q$ est un paramètre tel que $\chi(u) = -1$.

3.2.1 Encodage presque-injectif en genre $g = 3$

Dans cette sous-section, nous proposons un encodage presque-injectif sur des courbes hyperelliptiques de genre 3. Supposons que $\mathrm{char}(\mathbb{F}_q) = p \neq 2, 3, 7$. Soit $s \in \mathbb{F}_q^*$ tel que $31s^2 + 42s - 441 = 0$:
— Si $p \neq 31$ et $q \equiv 7 \bmod 8$, on a $\Delta_s = 56448 = 2^7 \times 3^2 \times 7^2$ qui est un carré, alors $s = \frac{-21 \pm 84\sqrt{2}}{31}$.
— Si $p = 31$ et $q \equiv 7 \bmod 8$, alors $s = \frac{21}{2}$.
Soit $\mathbb{H}_3 : y^2 = f_3(x) = x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$, avec $a_5 = sw^2,\ a_1 = \frac{sw^6}{3},\ a_3 = \frac{5sw^4}{3}$ et $a_0 = \frac{s-21}{21}w^7$,une courbe hyperelliptique de genre 3 sur $\mathbb{F}_q$ avec les conditions précédentes

sur q. On considère l'ensemble $\mathcal{R}_3 = \{r \in \mathbb{F}_q^*, f_3(w[ur^2(-651s-441)-1]) \neq 0\}$.

Algorithme 8 : Genus3-Encoding2

Entrées : La courbe hyperelliptique $\mathbb{H}_3$, un élément $r \in \mathcal{R}_3$
Sortie : Un point (x, y) sur $\mathbb{H}_3$

1 $v := w[ur^2(-651s-441)-1]$;
2 $\varepsilon := \chi(v^7 + a_5v^5 + a_3v^3 + a_1v + a_0)$;
3 $x := \frac{1+\varepsilon}{2}v + \frac{1-\varepsilon}{2}\left(\frac{w(-v+w)}{v+w}\right)$;
4 $y := -\varepsilon\sqrt{x^7 + a_5x^5 + a_3x^3 + a_1x + a_0}$;
5 **Retourner** (x, y).

Définition 3.2.1.

Dans la situation de l'algorithme 8, la fonction d'encodage pour la courbe hyperelliptique $\mathbb{H}_3$ est la fonction $\psi_3 : \mathcal{R}_3 \to \mathbb{H}_3 : r \mapsto \psi_3(r) = (x, y)$.

Théorème 3.2.2
L'algorithme 8 détermine un encodage presque-injectif déterministe $\psi_3 : \mathcal{R}_3 \to \mathbb{H}_3 : r \mapsto \psi_3(r) = (x, y)$, en temps $\mathcal{O}(\log^{2+\circ(1)} q)$, où $Z_3 = \mathbb{F}_q \backslash \mathcal{R}_3$ est un sous-ensemble de $\mathbb{F}_q$ d'au plus 15 éléments.

Preuve :

1. $f_3(v) \neq 0$ par définition de $\mathcal{R}_3$. Donc $\varepsilon = \chi(v^7 + a_5v^5 + a_3v^3 + a_1v + a_0) \neq 0$.
2. x est bien défini puisque $v + w = 0 \Leftrightarrow r = 0$ et $0 \notin \mathcal{R}_3$.
3. Montrons que $f_3(x)$ est un carré non nul.
   — Si $\varepsilon = 1$ i.e $f_3(v)$ est un carré non nul et $x = v$, ce qui montre donc que $f_3(x)$ est un carré non nul.
   — Si $\varepsilon = -1$ alors on a $x = \left(\frac{w(-v+w)}{v+w}\right)$ et $f_3(x) = \frac{1}{(v+\omega)^7}\Big[\omega^7(-v+\omega)^7 + a_5\omega^5(-v+\omega)^5(v+\omega)^2 + a_3\omega^3(-v+\omega)^3(v+\omega)^4 + a_1\omega(-v+\omega)(v+\omega)^6 + a_0(v+\omega)^7\Big]$.
   En utilisant $a_5 = sw^2$, $a_1 = \frac{sw^6}{3}$, $a_3 = \frac{5sw^4}{3}$ $a_0 = \frac{s-21}{21}w^7$, après quelques calculs, on obtient :

$$f_3(x) = \frac{(\frac{-62}{21}sw^7 - 2w^7)v^7 + (2sw^9 - 42w^9)v^5 + (\frac{10}{3}sw^{11} - 70w^{11})v^3 + (\frac{2}{3}sw^{13} - 14w^{13})v + \frac{64}{21}sw^{14}}{(v+w)^7}$$

On pose $\beta_7 = (\frac{-62}{21}sw^7 - 2w^7)$, $\beta_5 = (2sw^9 - 42w^9)$, $\beta_3 = (\frac{10}{3}sw^{11} - 70w^{11})$, $\beta_1 = (\frac{2}{3}sw^{13} - 14w^{13})$ et $\beta_0 = \frac{64}{21}sw^{14}$. Donc

$$f_3(x) = \frac{\beta_7}{(v+w)^7}\left[v^7 + \frac{\beta_5}{\beta_7}v^5 + \frac{\beta_3}{\beta_7}v^3 + \frac{\beta_1}{\beta_7}v + \frac{\beta_0}{\beta_7}\right]$$

On a $\frac{\beta_5}{\beta_7} = \frac{21(2sw^9 - 42w^9)}{-62sw^7 - 42w^7} = \frac{w^2(21s - 441)}{(-31s - 21s)}$. Puisque $31s^2 + 42s - 441 = 0$, alors $21s - 441 = -31s^2 - 21s$. On en déduit que $\frac{\beta_5}{\beta_7} = \frac{w^2(-31s^2 - 21s)}{-3ss - 21} = sw^2 = a_5$.

$\frac{\beta_3}{\beta_7} = \frac{(\frac{10}{3}sw^{11} - 70w^{11})}{(\frac{-62}{21}sw^7 - 2w^7)} = \frac{10sw^{11} - 210w^{11}}{\frac{3}{21}(-62sw^7 - 42w^7)} = \frac{5}{3}w^4\left(\frac{21(2s - 42)}{-62s - 42}\right)$
$= \frac{5}{3}w^4\left(\frac{(21s - 441)}{-31s - 21}\right) = \frac{5}{3}w^4\left(\frac{-31s^2 - 21}{-31s - 21}\right)$. Donc $\frac{\beta_3}{\beta_7} = \frac{5}{3}sw^4 = a_5$.

$\frac{\beta_1}{\beta_7} = \frac{(\frac{2}{3}sw^{13} - 14w^{13})}{(\frac{-62}{21}sw^7 - 2w^7)} = \frac{1}{3}\frac{21(2sw^{13} - 42w^{13})}{-62sw^7 - 42w^7} = \frac{w^6}{3}\frac{(21s - 441)}{-62s - 441} = \frac{w^6}{3}\frac{(-31s^2 - 21s)}{-31s - 21} = \frac{1}{3}sw^6 = a_1$.

$\frac{\beta_0}{\beta_7} = \frac{\frac{64}{21}sw^{14}}{(\frac{-62}{21}sw^7 - 2w^7)} = \frac{64sw^7}{-62s - 42} = \frac{32sw^7}{-31s - 21}$. Nous avons

$$\begin{aligned} 31s + 42s - 441 = 0 &\Leftrightarrow -31s - 42s + 441 = 0 \\ &\Leftrightarrow -31s^2 + 630s + 441 = 672s \\ &\Leftrightarrow (s - 21)(-31s - 21) = 32s * 21 \\ &\Leftrightarrow \frac{32sw^7}{-31s - 21} = \frac{s - 21}{21}w^7 \end{aligned}$$

Donc $\frac{\beta_0}{\beta_7} = \frac{32sw^7}{-31s - 21} = \frac{s-21}{21}w^7 = a_0$.
On a montré que $f_3(x) = \frac{\beta_7}{(v + w)^7}.f_2(v)$. Ce qui implique que $\chi(f_3(x)) = \chi(\beta_7)\chi(v + \omega))\chi(f_3(v)) = -\chi(\beta_7(v + w))$ puisque $f_3(v)$ est non nul et n'est pas un carré. Mais on sait que $\beta_7(v + w) = \omega^7\frac{-31s-21}{21}(wur^2(-651s - 441)) = uw^8r^2(-31s - 21)^2 = u\,(rw^4(-31s - 21))^2$ car $r \neq 0$ et est dans $\mathcal{R}_3$, Donc $\chi(\beta_7(v + w)) = \chi(u) = -1$. Ainsi $\chi(f_3(x)) = 1$, donc $f_3(x)$ est un carré non nul, on en déduit que $y = -\varepsilon\sqrt{f_3(x)}$ est bien défini.

□

Théorème 3.2.3
Dans la situation du Théorème 3.2.2 et de la Définition 3.2.1, on a

1. *Soit (x, y) un point de la courbe hyperelliptique $\mathbb{H}_3$, alors $(x, y) \in \mathrm{Im}(\psi_3)$ si et seulement si $uw(x + w)(-441 - 651s)$ est un carré non nul dans $\mathbb{F}_q$.*
2. *Soit $(x, y) \in \mathrm{Im}(\psi_3)$ et on définit $\bar{r}$ comme suit :*
   $\bar{r} = \sqrt{\frac{x + w}{uw(-441 - 651s)}}$ *si* $y \notin \sqrt{\mathbb{F}_q^2}$ *et* $\bar{r} = \sqrt{\frac{2w}{u(x + w)(-441 - 651s)}}$ *si* $y \in \sqrt{\mathbb{F}_q^2}$.

   Alors $\bar{r} \in \mathcal{R}_3$ et $\psi_3(\bar{r}) = (x, y)$.

Preuve :

1. — Supposons que $(x, y) \in \mathrm{Im}(\psi_3)$.
   — Si $\varepsilon = 1$ alors $x = v$. Donc $x+w = 0 \Leftrightarrow v+w = 0 \Leftrightarrow r = 0$ mais nous savons que $0 \notin \mathcal{R}_3$. On a donc $uw(x+w)(-441-651s) = uw(v+w)(-441-651s) = uw[ur^2w(-441-651s)](-441-651s) = u^2w^2r^4(-441-651s)^2$ qui est un carré non nul.

   — Si $\varepsilon = -1$ alors $x = \frac{w(-v+w)}{v+w}$, donc $x + w = 0 \Leftrightarrow \frac{-vw+w^2}{v+w} = -w \Leftrightarrow w = 0$ ce qui contredit le fait que $w \in \mathbb{F}_q^\star$. On a $uw(x+w)(-441-651s) = uw[\frac{w(-v+w)}{v+w} + w](-441-651s) = \frac{uw}{v+w}(2w^2)(-441-651s) = \frac{2w^2}{r^2}$, puisque $v + w = uwr^2(-441-651s)$. Comme 2 est un carré quand $q \equiv 7 \bmod 8$ alors $uw(x+w)(-441-651s)$ est un carré non nul.

   Nous concluons donc que dans tous les cas, $uw(x+w)(-441-651s)$ est un carré non nul dans $\mathbb{F}_q$.
   — Réciproquement, supposons que $uw(x+w)(-441-651s)$ est un carré non nul dans $\mathbb{F}_q$. On veut montrer que $(x, y) \in \mathrm{Im}(\psi_3)$. On pose $\bar{r} = \sqrt{\frac{x+w}{uw(-441-651s)}}$ si $y \notin \sqrt{\mathbb{F}_q^2}$ et $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-441-651s)}}$ si $y \in \sqrt{\mathbb{F}_q^2}$. Par les hypothèses précédentes $\dfrac{x+w}{uw(-441-651s)}$ et $\dfrac{2w}{u(x+w)(-441-651s)}$ sont bien définis et sont des carrés non nuls, D'où $\bar{r}$ est toujours bien défini.
   Nous allons prouver à présent que $\bar{r} \in \mathcal{R}_3$ et $(x, y) \in \mathrm{Im}(\psi_3)$. On définit $\bar{v}, \bar{\varepsilon}, \bar{x}, \bar{y}$ comme dans l'algorithme 8.
   Si $y \notin \sqrt{\mathbb{F}_q^2}$ alors $\bar{r} = \sqrt{\frac{x+w}{uw(-441-651s)}} \Longrightarrow \bar{v} = w[u\bar{r}^2(-441-651s) - 1] = w[\frac{x+w}{w} - 1] = x$. Donc on a $\bar{\varepsilon} = \chi(\bar{v}^7 + a_5\bar{v}^5 + a_3\bar{v}^3 + a_1\bar{v} + a_0) = \chi(f_3(\bar{v})) = 1$.
   Ainsi $\bar{x} = \frac{1+\bar{\varepsilon}}{2}\bar{v} + \frac{1-\bar{\varepsilon}}{2}\left(\frac{\omega(-\bar{v}+\omega)}{\bar{v}+\omega}\right) = \bar{v} = x$ et $\bar{y} = -\bar{\varepsilon}\sqrt{x^7 + a_5x^5 + a_3x^3 + a_1x + a_0} = -\sqrt{x^7 + a_5x^5 + a_3x^3 + a_1x + a_0} = y$.
   Si maintenant $y \in \sqrt{\mathbb{F}_q^2}$ alors $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-441-651s)}}$, puisque $\bar{v} = w[u\bar{r}^2(-441-651s) - 1]$ donc $\bar{v} = w\frac{w-x}{x+w}$.
   Après quelques calculs, on obtient : $\bar{\varepsilon} = \chi(\bar{v}^7 + a_5\bar{v}^5 + a_3\bar{v}^3 + a_1\bar{v} + a_0) = \chi\left(\left(\frac{\frac{-62}{21}sw^7 - 2w^7}{(x+\omega)^7}\right)(x^7 + a_5x^5 + a_3x^3 + a_1x + a_0)\right) = \chi\big(2 \times 21 \times w(x+w)(-31s-21)\big) = \chi(w(x+w)(-651s-441)) = -1$ car $uw(x+w)(-441-651s)$ est un carré non nul et $\chi(u) = -1$. Comme $\bar{\varepsilon} = -1$ alors $\bar{x} = \frac{\omega(-\bar{v}+\omega)}{\bar{v}+\omega} = x$ et $\bar{y} = -\bar{\varepsilon}\sqrt{x^7 + a_5x^5 + a_3x^3 + a_1x + a_0} = \sqrt{x^7 + a_5x^5 + a_3x^3 + a_1x + a_0} = y$.
   Dans les deux cas, nous avons $\bar{x} = x$ et $\bar{y} = y$. Comme pour tout $\bar{v}$, on a $\bar{v}^7 + a_5\bar{v}^5 + a_3\bar{v}^3 + a_1\bar{v} + a_0 \neq 0$, donc $f_3(\bar{v}) \neq 0$, ainsi $\bar{r} \in \mathcal{R}_3$.
2. Découle de la preuve précédente.

□

3.2.2 Encodage presque-injectif en genre $g = 4$

De façon similaire à la sous-section précédente, dans cette sous-section, nous proposons un encodage presque-injectif sur des courbes hyperelliptiques de genre 4.

Nous supposons que $\text{char}(\mathbb{F}_q) = p \neq 2, 3$. Soit $s \in \mathbb{F}_q^*$ tel que $127s^2 + 72s - 1296 = 0$:

— Si $p \neq 127$ et $q \equiv 7 \bmod 8$, on a $\Delta_s = 663552 = 2^{13} \times 3^4$ qui est un carré, alors $s = (-36 \pm 288\sqrt{2})/127$.

— Si $p = 127$ et $q \equiv 7 \bmod 8$, alors $s = 18$.

Soit $\mathbb{H}_4 : y^2 = f_4(x) = x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$, avec $a_7 = sw^2$, $a_5 = \frac{7sw^4}{2}$, $a_3 = \frac{7sw^6}{3}$, $a_1 = \frac{sw^8}{4}$, $a_0 = \frac{s-36}{36}w^9$ une courbe hyperelliptique de genre 4 sur $\mathbb{F}_q$ avec les conditions précédentes sur q. On pose
$\mathcal{R}_4 = \{r \in \mathbb{F}_q^*, f_4(w[ur^2(-2286s - 648) - 1]) \neq 0\}$.

Algorithme 9 : Genus4-Encoding3

Entrées : La courbe hyperelliptique $\mathbb{H}_4$, un élément $r \in \mathcal{R}_4$
Sortie : Un point (x, y) sur $\mathbb{H}_4$

1 $v := w[ur^2(-2286s - 648) - 1]$;
2 $\varepsilon := \chi(v^9 + a_7v^7 + a_5v^5 + a_5v^5 + a_3v^3 + a_1v + a_0)$;
3 $x := \frac{1+\varepsilon}{2}v + \frac{1-\varepsilon}{2}\left(\frac{w(-v+w)}{v+w}\right)$;
4 $y := -\varepsilon\sqrt{x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0}$;
5 **Retourner** (x, y).

Définition 3.2.4.

Dans la situation de l'Algorithme 9, la fonction d'encodage pour la courbe hyperelliptique $\mathbb{H}_4$ est la fonction $\psi_4 : \mathcal{R}_4 \to \mathbb{H}_4 : r \mapsto \psi_4(r) = (x, y)$.

Théorème 3.2.5
L'algorithme 9 détermine un encodage déterministe presque-injectif $\psi_4 : \mathcal{R}_4 \to \mathbb{H}_4 : r \mapsto \psi_4(r) = (x, y)$, en temps $\mathcal{O}(\log^{2+\circ(1)} q)$, où $Z_4 = \mathbb{F}_q \backslash \mathcal{R}_4$ est un sous-ensemble de $\mathbb{F}_q$ d'au plus 19 éléments.

Preuve :

1. $f_4(v) \neq 0$ par définition de $\mathcal{R}_4$. Donc $\varepsilon = \chi(v^9+a_7v^7+a_5v^5+a_5v^5+a_3v^3+a_1v+a_0) \neq 0$
2. x est bien défini puisque $v + w = 0 \Leftrightarrow r = 0$ et $0 \notin \mathcal{R}_4$.
3. Prouvons que $f_4(x)$ est un carré non nul. Pour cela, on va considérer les cas suivant : $\varepsilon = 1$ ou $\varepsilon = -1$.
   — Si $\varepsilon = 1$ i.e $f_4(v)$ est un carré non nul et $x = v$ alors $f_4(x)$ est un carré non nul.
   — Si $\varepsilon = -1$ alors on a $x = \left(\frac{w(-v+w)}{v+w}\right)$ et $f_4(x) = \frac{g_4(v)}{(v+\omega)^9}$ avec $g_4(v) = \omega^9(-v+\omega)^9 + a7\omega^7(-v+\omega)^7(v+\omega)^2 + a_5\omega^5(-v+\omega)^5(v+\omega)^4 + a_3\omega^3(-v+\omega)^3(v+\omega)^6 + a_1\omega(-v+\omega)(v+\omega)^8 + a_0(v+\omega)^7$.
   En utilisant le fait que $a_7 = sw^2$, $a_1 = \frac{sw^8}{4}$, $a_3 = \frac{7sw^6}{3}$, $a_5 = \frac{7sw^4}{2}$ $a_0 = \frac{s-36}{36}w^9$, après quelques calculs, on obtient :

$f_4(x) = \dfrac{\gamma_9 v^9 + \gamma_7 v^7 + \gamma_5 v^5 + \gamma_3 v^3 + \gamma_1 v + \gamma_0}{(v+w)^9}$ avec $\gamma_9 = -\frac{127}{18}sw^9 - 2w^9$; $\gamma_7 = 2sw^{11} - 72w^{11}$; $\gamma_5 = 7sw^{13} - 252w^{13}$; $\gamma_3 = \frac{14}{3}sw^{15} - 168w^{15}$; $\gamma_1 = \frac{1}{2}sw^{17} - 18w^{17}$ et $\gamma_0 = \frac{64}{9}sw^{18}$.

Cela montre que $f_4(x) = \dfrac{\gamma_9}{(v+w)^9}\left(v^9 + \dfrac{\gamma_7}{\gamma_9}v^7 + \dfrac{\gamma_5}{\gamma_9}v^5 + \dfrac{\gamma_3}{\gamma_9}v^3 + \dfrac{\gamma_1}{\gamma_9}v + \dfrac{\gamma_0}{\gamma_9}\right)$.
Maintenant, prouvons que $\dfrac{\gamma_7}{\gamma_9} = a_7$, $\dfrac{\gamma_5}{\gamma_9} = a_5$, $\dfrac{\gamma_3}{\gamma_9} = a_3$, $\dfrac{\gamma_1}{\gamma_9} = a_1$ et $\dfrac{\gamma_0}{\gamma_9} = a_0$.

$\dfrac{\gamma_7}{\gamma_9} = \dfrac{2sw^{11} - 72w^{11}}{-\frac{127}{18}sw^9 - 2w^9} = \dfrac{(36s - 1296)w^2}{-127s - 36}w^2$. On a $127s^2 + 72s - 1296 = 0 \Leftrightarrow 36s - 1296 = -127s^2 - 36s$; ceci implique que $\dfrac{\gamma_7}{\gamma_9} = \dfrac{(-127s^2 - 36s)}{-127s - 36}w^2 = sw^2$.
$\dfrac{\gamma_5}{\gamma_9} = \dfrac{7sw^{13} - 252w^{13}}{-\frac{127}{18}sw^9 - 2w^9} = \dfrac{7(18s - 648)}{-127s - 36}w^4 = \frac{7}{2}\dfrac{(36s - 1296)}{-127s - 36}w^4 = \frac{7}{2}\dfrac{-127s^2 - 36s}{-127s - 36}w^4 = \frac{7}{2}sw^4 = a_5$.
$\dfrac{\gamma_3}{\gamma_9} = \dfrac{\frac{14}{3}sw^{15} - 168w^{15}}{-\frac{127}{18}sw^9 - 2w^9} = \dfrac{6(14s - 504)}{-127s - 36}w^6 = \dfrac{7(12s - 432)}{-127s - 36}w^6 = \frac{7}{3}\dfrac{36s - 1296}{-127s - 36}w^6 = \frac{7}{3}\dfrac{-127s^2 - 36s}{-127s - 36} = \frac{7}{3}sw^6 = a_3$.

$\dfrac{\gamma_1}{\gamma_9} = \dfrac{\frac{1}{2}sw^{17} - 18w^{17}}{-\frac{127}{18}sw^9 - 2w^9} = \dfrac{9(s - 36)}{-127s - 36}w^8 = \frac{1}{4}\dfrac{36s - 1296}{-127s - 36}w^8 = \dfrac{sw^8}{4} = a_1$. $\dfrac{\gamma_0}{\gamma_9} = \dfrac{\frac{64}{9}sw^{18}}{-\frac{127}{18}sw^9 - 2w^9} = \dfrac{128sw^9}{-127s - 36}$. En utilisant le fait que $127s^2 + 72s - 1296 = 0$, on va montrer à présent que $\frac{128s}{-127s-36} = \frac{s-36}{36}$.

$$\begin{aligned} 127s^2 + 72s - 1296 = 0 &\Leftrightarrow 4608s = -127s^2 + 4536s + 1296 \\ &\Leftrightarrow 36 \times 128s = (s - 36)(-127s - 36) \\ &\Leftrightarrow \frac{128s}{-127s - 36} = \frac{s - 36}{36} \end{aligned}$$

Donc $\dfrac{\gamma_0}{\gamma_9} = \dfrac{128sw^9}{-127s - 36} = \dfrac{s - 36}{36}w^9 = a_0$.

On a montré que $f_4(x) = \dfrac{\gamma_9}{(v+w)^9}.f_4(v)$. Et cela entraine donc que $\chi(f_4(x)) = \chi(\gamma_9)\chi(v + \omega))\chi(f_4(v)) = -\chi(\gamma_9(v + w))$ car $f_4(v)$ est un non nul et n'est pas un carré. Nous savons que $\gamma_9(v + w) = \omega^9\frac{-127s-36}{18}(wur^2(-2286s - 648)) = uw^{10}r^2(-127s - 36)^2 = u\left(rw^5(-127s - 36)\right)^2$. Puisque $r \neq 0$ dans $\mathcal{R}_4$, alors $\chi(\gamma_9(v + w)) = \chi(u) = -1$. Ainsi $\chi(f_4(x)) = 1$, par conséquent $f_4(x)$ est un carré non nul, on en déduit que $y = -\varepsilon f_4(x)$ est bien défini.

□

Théorème 3.2.6
Dans la situation du Théorème 3.2.5 et de la Définition 3.2.4, nous avons les propriétés suivantes.

1. *Soit* (x, y) *un point de la courbe hyperelliptique* $\mathbb{H}_4$. *Alors* $(x, y) \in \mathrm{Im}(\psi_4)$ *si et seulement si* $uw(x + w)(-2286s - 648)$ *est un carré non nul dans* $\mathbb{F}_q$.
2. *Soit* $(x, y) \in \mathrm{Im}(\psi_4)$. *On définit* $\bar{r}$ *comme suit :* $\bar{r} = \sqrt{\frac{x+w}{uw(-2286s-648)}}$ *if* $y \notin \sqrt{\mathbb{F}_q^2}$ *et* $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-2286s-648)}}$ *et* $y \in \sqrt{\mathbb{F}_q^2}$. *Alors* $\bar{r} \in \mathcal{R}_4$ *et* $\psi_4(\bar{r}) = (x, y)$.

Preuve : Similaire à la preuve du Théorème 3.2.3.

□

3.2.3 Encodage presque-injectif en genre $g = 5$

Nous proposons ici un encodage presque-injectif sur des courbes hyperelliptiques de genre 5.
Nous supposons que $\mathrm{char}(\mathbb{F}_q) = p \neq 2, 5, 11$. Soit $s \in \mathbb{F}_q^*$ tel que $511s^2 + 110s - 3025 = 0$:

— Si p ne divise pas 511 et que $q \equiv 7 \bmod 8$, on a $\Delta_s = 6195200 = 2^{11} \times 5^2 \times 11^2$ qui est un carré, alors $s = \dfrac{-55 \pm 880\sqrt{2}}{511}$.

— Si maintenant p divise 511 et $q \equiv 7 \bmod 8$, alors $s = \frac{55}{2}$.

Soit $\mathbb{H}_5 : y^2 = f_5(x) = x^{11} + a_9x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$, avec $a_9 = sw^2$, $a_7 = 6sw^4$, $a_5 = \dfrac{42sw^6}{5}$, $a_3 = 3sw^8$, $a_1 = \dfrac{sw^{10}}{5}$ et $a_0 = \dfrac{s - 55}{55}w^{11}$, une courbe hyperelliptique de genre 5 sur $\mathbb{F}_q$ avec les conditions précédentes sur q. On pose

$$\mathcal{R}_5 = \left\{r \in \mathbb{F}_q^*, f_5(w[ur^2(-28105s - 3025) - 1]) \neq 0\right\}$$

Algorithme 10 : Genus5-Encoding4

Entrées : La courbe hyperelliptique $\mathbb{H}_5$, un élément $r \in \mathcal{R}_5$
Sortie : Un point (x, y) sur $\mathbb{H}_5$

1 $v := w[ur^2(-28105s - 3025) - 1]$;
2 $\varepsilon := \chi(v^9 + a_9v^9 + a_7v^7 + a_5v^5v^3 + a_5v^5 + a_1v + a_0)$;
3 $x := \dfrac{1 + \varepsilon}{2}v + \dfrac{1 - \varepsilon}{2}\left(\dfrac{w(-v + w)}{v + w}\right)$;
4 $y := -\varepsilon\sqrt{x^{11} + a_9x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0}$;
5 **Retourner** (x, y).

Définition 3.2.7.

Dans la situation de l'algorithme 10, la fonction d'encodage pour la courbe hyperelliptique $\mathbb{H}_5$ est la fonction $\psi_5 : \mathcal{R}_5 \to \mathbb{H}_5 : r \mapsto \psi_5(r) = (x, y)$.

Théorème 3.2.8
L'algorithme 10 détermine un encodage déterministe presque-injectif $\psi_5 : \mathcal{R}_5 \to \mathbb{H}_5 : r \mapsto \psi_5(r) = (x, y)$, *en temps* $\mathcal{O}(\log^{2+\circ(1)} q)$, *où* $Z_5 = \mathbb{F}_q \backslash \mathcal{R}_5$ *est un sous-ensemble de* $\mathbb{F}_q$ *d'au plus* 23 *éléments.*

Preuve :

1. $f_5(v) \neq 0$ par définition de $\mathcal{R}_5$. Donc $\varepsilon = \chi(v^9 + a_9v^9 + a_7v^7 + a_5v^5v^3 + a_5v^5 + a_1v + a_0) \neq 0$
2. x est bien défini puisque $v + w = 0 \Leftrightarrow r = 0$ et $0 \notin \mathcal{R}_5$
3. Prouvons que $f_5(x)$ est un carré non nul.
   — Si $\varepsilon = 1$ i.e $f_5(v)$ un carré non nul et $x = v$ alors $f_5(x)$ est un carré non nul.
   — Si $\varepsilon = -1$ alors $x = \left(\frac{w(-v+w)}{v+w}\right)$ et $f_5(x) = \dfrac{g_5(v)}{(v+\omega)^{11}}$ avec $g_5(v) = \omega^{11}(-v + \omega)^{11} + a_9\omega^9(-v + \omega)^9(v + \omega)^2 + a_7\omega^7(-v + \omega)^7(v + \omega)^4 + a_5\omega^5(-v + \omega)^5(v + \omega)^6 + a_3\omega^3(-v + \omega)^3(v + \omega)^8 + a_1\omega(-v + \omega)(v + \omega)^{10} + a_0(v + \omega)^{11}$.
   On sait que $a_9 = sw^2$, $a_7 = 6sw^4$, $a_5 = \dfrac{42sw^6}{5}$, $a_3 = 3sw^8$, $a_1 = \dfrac{sw^{10}}{5}$, $a_0 = \dfrac{s-55}{55}w^{11}$, après quelques calculs, on a :

   $$f_5(x) = \frac{\eta_{11}v^{11} + \eta_9v^9 + \eta_7v^7 + \eta_5v^5 + \eta_3v^3 + \eta_1v + \eta_0}{(v+w)^9} \text{ avec}$$

   $\eta_{11} = -\frac{1022}{55}sw^{11} - 2w^{11}$, $\eta_9 = 2sw^{13} - 110w^{13}$; $\eta_7 = 12sw^{15} - 660w^{15}$; $\eta_5 = \frac{84}{5}sw^{17} - 924w^{17}$; $\eta_3 = 6sw^{19} - 330w^{19}$; $\eta_1 = \frac{2}{5}sw^{21} - 22w^{21}$ et $\eta_0 = \frac{1024}{55}sw^{22}$.
   L'expression de la fonction f_5 devient :
   $f_5(x) = \dfrac{\eta_{11}}{(v+w)^{11}}\left(v^{11} + \frac{\eta_9}{\eta_{11}}v^9 + \frac{\eta_7}{\eta_{11}}v^7 + \frac{\eta_5}{\eta_{11}}v^5 + \frac{\eta_3}{\eta_{11}}v^3 + \frac{\eta_1}{\eta_{11}}v + \eta_0\right)$.

   Notre principal objectif maintenant est de prouver que $\frac{\eta_9}{\eta_{11}} = a_9$; $\frac{\eta_7}{\eta_{11}} = a_7$; $\frac{\eta_5}{\eta_{11}} = a_5$; $\frac{\eta_3}{\eta_{11}} = a_3$; $\frac{\eta_1}{\eta_{11}} = a_1$ et $\frac{\eta_0}{\eta_{11}} = a_0$.
   $\frac{\eta_9}{\eta_{11}} = \dfrac{2sw^{13} - 110w^{13}}{-\frac{1022}{55}sw^{11} - 2w^{11}} = \dfrac{55s - 3025}{-511s - 55}w^2$. Comme $511s^2 + 110s - 3025 = 0$, on a $55s - 3025 = 511s^2 - 55s$. Et donc on a $\dfrac{\eta_9}{\eta_{11}} = \dfrac{-511s^2 - 55s}{-511s - 55}w^2 = sw^2 = a_9$. $\dfrac{\eta_7}{\eta_{11}} = \dfrac{12sw^{15} - 660w^{15}}{-\frac{1022}{55}sw^{11} - 2w^{11}} = \dfrac{55(6s - 55)}{-511s - 55}w^4 = \dfrac{6(55s - 3025)}{-511s - 55}w^4 = 6 \times \dfrac{-511s^2 - 55s}{-511s - 55}w^4 = 6sw^4 = a_7$.

   $$\frac{\eta_5}{\eta_{11}} = \frac{\frac{84}{5}sw^{17} - 924w^{17}}{-\frac{1022}{55}sw^{11} - 2w^{11}} = \frac{55}{5}.\frac{42s - 2310}{-511s - 55}w^6$$

   $$= \frac{42}{5}.\frac{55s - 3025}{-511s - 55}w^6 = \frac{42}{5}.\frac{-511s^2 - 55s}{-511s - 55}w^6 = \frac{42}{5}sw^6 = a_5.$$

   $$\frac{\eta_3}{\eta_{11}} = \frac{6sw^{19} - 330w^{19}}{-\frac{1022}{55}sw^{11} - 2w^{11}} = \frac{55(3s - 165)}{-511s - 55}w^8 = \frac{3(55s - 3025)}{-511s - 55}w^8 = a_3.$$

   $$\frac{\eta_1}{\eta_{11}} = \frac{\frac{2}{5}sw^{21} - 22w^{21}}{-\frac{1022}{55}sw^{11} - 2w^{11}} = \frac{55}{5} \times \frac{s - 55}{-511s - 210}w^{10} = \frac{sw^{10}}{5} = a_1.$$

$$\frac{\eta_0}{\eta_{11}} = \frac{\frac{1024}{55}sw^{22}}{-\frac{1022}{55}sw^{11} - 2w^{11}} = \frac{512s}{-511s - 55}w^{11}.$$

$$\begin{aligned} 511s^2 + 110s - 3025 = 0 &\Leftrightarrow 28160s = -511s^2 + 28050s + +3025 \\ &\Leftrightarrow 55 \times 512s = (s-55)(-511s-55) \\ &\Leftrightarrow \frac{512s}{-511s-55} = \frac{s-55}{55} \end{aligned}$$

Donc $\frac{\eta_0}{\eta_{11}} = \frac{512s}{-511s-55}w^{11} = \frac{s-55}{55}w^{11} = a_0$.
On a montré que $f_5(x) = \frac{\eta_{11}}{(v+w)^{11}}.f_5(v)$. Ce qui entraine que $\chi(f_5(x)) = \chi(\eta_{11})\chi(v+\omega))\chi(f_5(v)) = -\chi(\eta_{11}(v+w))$ car $f_5(v)$ est un non nul et n'est pas un carré. Mais, nous avons $\eta_{11}(v+w) = \eta_{11}\frac{-127s-36}{18}(wur^2(-28105s-3025)) = uw^{10}r^2(-511s-55)^2 = u\left(rw^6(-511s-55)\right)^2$. Puisque $r \neq 0$ dans $\mathcal{R}_5$, alors $\chi(\eta_{11}(v+w)) = \chi(u) = -1$. Ainsi $\chi(f_5(x)) = 1$, tdonc $f_5(x)$ est un carré non nul, on en déduit que $y = -\varepsilon\sqrt{f_5(x)}$ est bien défini.

□

Théorème 3.2.9
Dans la situation du Théorème 3.2.8 et de la Définition 3.2.7, nous avons les propriétés suivantes.

1. *Soit (x,y) un point de la courbe hyperelliptique $\mathbb{H}_5$, alors $(x,y) \in \mathrm{Im}(\psi_5)$ si et seulement si $uw(x+w)(-28105s-3025)$ est un carré non nul dans $\mathbb{F}_q$.*
2. *Soit $(x,y) \in \mathrm{Im}(\psi_5)$ et on définit $\bar{r}$ comme suit :*
   $\bar{r} = \sqrt{\frac{x+w}{uw(-28105s-3025)}}$ *si* $y \notin \sqrt{\mathbb{F}_q^2}$ *et* $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-28105s-3025)}}$ *si* $y \in \sqrt{\mathbb{F}_q^2}$. *Alors* $\bar{r} \in \mathcal{R}_5$ *et* $\psi_5(\bar{r}) = (x,y)$.

Preuve : Similaire à la preuve du Théorème 3.2.3.

□

3.2.4 Encodage presque-injectif en genre $g = 1$ utilisant notre technique

Dans cette sous-section, nous montrons comment on peut encoder sur une courbe elliptique donnée par $E = \mathbb{H}_1 : y^2 = x^3 + a_1x + a_0$ sur un corps fini $\mathbb{F}_q$ où $q \equiv 7 \bmod 8$ en utilisant la même approche que dans les sous-sections précédentes.

On suppose que $\mathrm{char}(\mathbb{F}_q) = p \neq 2,3$. Soit $s \in \mathbb{F}_q^*$ tel que $s^2 + 6s - 9 = 0$. Comme $q \equiv 7 \bmod 8$, on a $\Delta_s = 72 = 2^3 \times 3^2$ qui est un carré, alors $s = -3 \pm 3\sqrt{2}$. Soit $E = \mathbb{H}_1 : y^2 = f_1(x) = x^3 + a_1x + a_0$, avec $a_1 = sw^2$ et $a_0 = \frac{s-3}{3}w^3$ une courbe elliptique sur $\mathbb{F}_q$ avec

les conditions précédentes sur q. On pose $\mathcal{R}_1 = \{r \in \mathbb{F}_q^*, f_1(w[ur^2(-3s-9)-1]) \neq 0\}$.

Algorithme 11 : Genus1-Encoding5

Entrées : La courbe elliptique $E = \mathbb{H}_1$, un élément $r \in \mathcal{R}_1$
Sortie : Un point (x, y) de E

1 $v := w[ur^2(-3s-9)-1]$;
2 $\varepsilon := \chi(v^3 + a_1 v + a_0)$;
3 $x := \frac{1+\varepsilon}{2} v + \frac{1-\varepsilon}{2}\left(\frac{w(-v+w)}{v+w}\right)$;
4 $y := -\varepsilon\sqrt{x^3 + a_1 x + a_0}$;
5 **Retourner** (x, y).

Définition 3.2.10.

Dans la situation de l'Algorithme 11, la fonction d'encodage pour la courbe elliptique E est la fonction $\psi_1 : \mathcal{R}_1 \to E : r \mapsto \psi_1(r) = (x, y)$.

Théorème 3.2.11
L'algorithme 11 détermine un encodage déterministe presque-injectif $\psi_1 : \mathcal{R}_1 \to E : r \mapsto \psi_1(r) = (x, y)$, en temps $\mathcal{O}(\log^{2+\circ(1)} q)$, où $Z_1 = \mathbb{F}_q \backslash \mathcal{R}_1$ est un sous-ensemble de $\mathbb{F}_q$ d'au plus 9 éléments.

Preuve :

1. $f_1(v) \neq 0$ par définition de $\mathcal{R}_1$. Donc $\varepsilon = \chi(v^3 + a_1 v + a_0) \neq 0$
2. x est bien défini puisque $v + w = 0 \Leftrightarrow r = 0$ et $0 \notin \mathcal{R}_1$.
3. Montrons que $f_1(x)$ est un carré non nul.
   — Si $\varepsilon = 1$ i.e $f_1(v)$ est un carré non nul et $x = v$ alors $f_1(x)$ est un carré non nul.
   — Si $\varepsilon = -1$ alors on a $x = \left(\frac{w(-v+w)}{v+w}\right)$ et $f_1(x) = \frac{1}{(v+\omega)^3}\Big[\omega^3(-v+\omega)^3 + a_1\omega(-v+\omega)(v+\omega)^2 + a_0(v+\omega)^3\Big]$. On sait que $a_1 = sw^2$, $a_0 = \frac{s-3}{3}w^3$, donc
   $$f_1(x) = \frac{(-2sw^3/3 - 2w^3)v^3 + (2sw^5 - 6w^5)v + 4sw^6/3}{(v+\omega)^3}$$
   $$f_1(x) = \frac{(-2sw^3/3 - 2w^3)}{(v+\omega)^3}\left(v^3 + \frac{(2sw^5 - 6w^5)}{(-2sw^3/3 - 2w^3)}v + \frac{4sw^6/3}{(-2sw^3/3 - 2w^3)}\right).$$
   On a $\frac{(2sw^5 - 6w^5)}{(-2sw^3/3 - 2w^3)} = \frac{3s-9}{-s-3}w^2 = sw^2 = a_1$ car $3s - 9 = -s^2 - 3s$ et $\frac{4sw^6/3}{(-2sw^3/3 - 2w^3)} = \frac{s-3}{3}w^3 = a_0$ puisque $s^2 + 6s - 9 = 0 \Leftrightarrow 6s = -s^2 + 9 \Leftrightarrow \frac{2s}{-s-3} = \frac{s-3}{3}$. Nous avons montré que $f_1(x) = \frac{(-2sw^3/3 - 2w^3)}{(v+\omega)^3}.f_1(v)$. Ce qui implique que $\chi(f_1(x)) = \chi(-2sw^3/3 - 2w^3)\chi(v+\omega))\chi(f_1(v)) = -\chi(w(-3s-9s)(v+w))$ car $f_1(v)$ est un non nul et n'est pas un carré. On a $w(-3s-9s)(v+w) = w(-3s-9s)(wur^2(-3s-9) = uw^2r^2(-3s-9s)^2 = u\,(rw(-3s-9))^2$.

Comme $r \neq 0$ et est dans $\mathcal{R}_1$, alors $\chi(w(-3s-9s)(v+w)) = \chi(u) = -1$. Ainsi $\chi(f_1(x)) = 1$, donc $f_1(x)$ est un carré non nul, on en déduit que $y = -\varepsilon\sqrt{f_1(x)}$ est bien défini.

□

Remarque 3.2.12
Nous soulignons que nous avons construit une fonction d'encodage sur la courbe elliptique $E = \mathbb{H}_1 : y^2 = f_1(x) = x^3 + a_1x + a_0$ *avec* $a_1 = sw^2$ $a_0 = \dfrac{s-3}{3}w^3$ *dans le but de trouver des formules unifiées en genre* $g \leqslant 5$ *même si la famille de ces courbes elliptiques est peut-être petite.*

Théorème 3.2.13
Dans la situation du Théorème 3.2.11 et de la Définition 3.2.10, nous avons :

1. *Soit* (x, y) *un point de la courbe elliptique* E*, alors* $(x, y) \in \mathrm{Im}(\psi_1)$ *si et seulement si* $uw(x + w)(-3s - 9)$ *est un carré non nul dans* $\mathbb{F}_q$*.*
2. *Soit* $(x, y) \in \mathrm{Im}(\psi_1)$ *et déffinissons* $\bar{r}$ *comme suit :* $\bar{r} = \sqrt{\dfrac{x+w}{uw(-3s-9)}}$ *si* $y \notin \sqrt{\mathbb{F}_q^2}$ *et* $\bar{r} = \sqrt{\dfrac{2w}{u(x+w)(-3s-9)}}$ *si* $y \in \sqrt{\mathbb{F}_q^2}$*. Alors* $\bar{r} \in \mathcal{R}_1$ *et* $\psi_1(\bar{r}) = (x, y)$*.*

Preuve : Similaire à la preuve du théorème 3.2.3.

□

3.3 Formules unifiées pour les cinq encodages

Notre but principal dans cette section est de trouver des formules afin d'unifier les cinq algorithmes [4, 8, 9, 10, 11] et les théorèmes [2.3.10, 3.2.3, 3.2.6, 3.2.9, 3.2.13].

Genre g	Courbe
1	$\mathbb{H}_1 = E : y^2 = f_1(x) = x^3 + a_1x + a_0$
2	$\mathbb{H}_2 : y^2 = f_1(x) = x^5 + a_3x^3 + a_1x + a_0$
3	$\mathbb{H}_3 : y^2 = f_2(x) = x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$
4	$\mathbb{H}_4 : y^2 = f_3(x) = x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$
5	$\mathbb{H}_5 : y^2 = f_4(x) = x^{11} + a_9x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$

TABLE 3.1 – **Les différentes familles de courbes hyperelliptiques considérées dans ce chapitre**

Genre g	Équation du second degré	Valeur de v
1	$s^2+6s-9=0$	$v=w[ur^2(-3s-9)-1]$
2	$7s^2+20s-100=0$	$v=w[ur^2(-35s-50)-1]$
3	$31s^2+42s-441=0$	$v=w[ur^2(-651s-441)-1]$
4	$127s^2+72s-1296=0$	$v=w[ur^2(-2286s-648)-1]$
5	$511s^2+110s-3025=0$	$v=w[ur^2(-28105s-3025)-1]$

TABLE 3.2 – **L'équation fondamentale du second degré et les différentes valeurs de v définies dans les algorithmes 4, 8, 9, 10 et 11.**

Remarque 3.3.14

1. *Le paramètre $s\in\mathbb{F}_q^*$ satisfait l'équation $\alpha_g s^2+\beta_g s-\gamma_g=0$ où $\alpha_g=2^{2g-1}-1$; $\beta_g=2\times g\times\deg(f_g)=4g^2+2g$ et $\gamma_g=(g\times\deg(f_g))^2=(2g^2+g)^2$.*
2. *Il faut noter que $\alpha_g=2^{2g-1}-1$ est un nombre de Mersenne et que $M_2=3$, $M_3=7$, $M_4=127$ sont respectivement les deuxième, troisième et quatrième nombres premiers de Mersenne.*

Remarque 3.3.15

$v=v(g)=w[ur^2(-m_g s-n_g)-1]$ *où*

— $m_g=\dfrac{1}{2}\times\alpha_g\times\beta_g$ *si g est impair et* $m_g=\dfrac{1}{4}\times\alpha_g\times\beta_g$ *si g est pair.*

— $n_g=(g\times\deg(f_g))^2=(2g^2+g)^2$ *si le genre g est impair et* $n_g=\dfrac{1}{2}\times(g\times\deg(f_g))^2=\dfrac{1}{2}\times(2g^2+g)^2$ *si g est pair.*

Genre g	a_1	$a_{(2g-1)}$	a_0	$a_3, g\geqslant 3$
1	$\frac{sw^2}{1}$	sw^2	$\frac{s-3}{3}w^3$	-
2	$a_1=\frac{sw^4}{2}$	sw^2	$\frac{s-10}{10}w^5$	-
3	$a_1=\frac{sw^6}{3}$	sw^2	$\frac{s-21}{21}w^7$	$\frac{5sw^4}{3}$
4	$a_1=\frac{sw^8}{4}$	sw^2	$\frac{s-36}{36}w^9$	$\frac{7sw^6}{3}$
5	$a_1=\frac{sw^{10}}{5}$	sw^2	$\frac{s-55}{55}w^{11}$	$3sw^8=\frac{9}{3}sw^8$

TABLE 3.3 – **La relation entre le genre g et les valeurs des coefficients a_0, a_1, $a_{(2g-1)}$ et a_3 dans les différentes courbes hyperelliptiques $\mathbb{H}_g$.**

Remarque 3.3.16

Notez que pour tout genre $g\leqslant 5$, $a_0=\dfrac{\left(s-(2g^2+g)\right)}{(2g^2+g)}w^{(2g+1)}$, $a_1=\frac{sw^{2g}}{g}$, $a_{(2g-1)}=sw^2$ *et* $a_3=\frac{2g-1}{3}sw^{2g-2}$ *pour $g\geqslant 3$.*

Encodage presque-injectif sur $\mathbb{H}_g$

Soit $g \in \{1,2,3,4,5\}$. On suppose que $\text{char}(\mathbb{F}_q) = p$, $p \neq 2$ et $p \nmid (2g^2+g)$. Soit $s \in \mathbb{F}_q^*$ tel que $\alpha_g s^2 + \beta_g s - \gamma_g = 0$ où $\alpha_g = 2^{2g-1}-1$; $\beta_g = 4g^2+2g$ et $\gamma_g = (2g^2+g)^2$. Si p divise α_g alors $s = \gamma_g/\beta_g$. Si maintenant $p \nmid \alpha_g$; on a $\Delta_s = \beta_g^2 + 4 \times \alpha_g \times \gamma_g$ et alors $s = (-\beta_g \pm \sqrt{\Delta_s})/(2 * \alpha_g)$.

Soit $\mathbb{H}_g : y^2 = h_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1 x + a_0$ une courbe hyperelliptique de genre g sur $\mathbb{F}_q$ avec les conditions précédentes sur q où $a_0 = \frac{\left(s-(2g^2+g)\right)}{(2g^2+g)} w^{(2g+1)}$, $a_{(2g-1)} = sw^2$, $a_1 = \frac{sw^{2g}}{g}$ pour $g \geqslant 1$, $a_3 = \frac{2g-1}{3} sw^{2g-2}$ pour $g \geqslant 3$, $a_5 = \dfrac{7sw^4}{2}$ si $g = 4$ et $a_7 = 6sw^4$, $a_5 = \dfrac{42sw^6}{5}$ si $g = 5$. On définit

- $\mathcal{R}_g = \left\{ r \in \mathbb{F}_q^*, h_g(w[ur^2(-m_g s - n_g) - 1]) \neq 0 \right\}$;
- $m_g = \frac{1}{2} \times \alpha_g \times \beta_g$ si le genre g est impair et $m_g = \frac{1}{4} \times \alpha_g \times \beta_g$ si g est pair quand $\alpha_g = 2^{2g-1}-1$ et $\beta_g = 4g^2+2g$;
- $n_g = (2g^2+g)^2$ si g est impair et $n_g = \dfrac{1}{2} \times (2g^2+g)^2$ si g est pair.

L'algorithme suivant généralise les algorithmes 4, 8, 9, 10 et 11.

Algorithme 12 : Genus-g-Encoding6-Generalization

Entrées : La courbe hyperelliptique $\mathbb{H}_g$, un élément $r \in \mathcal{R}_g$
Sortie : Un point (x,y) de $\mathbb{H}_g$

1 $v := v(g) = w[ur^2(-m_g s - n_g) - 1]$;
2 $\varepsilon := \chi(v^{(2g+1)} + a_{(2g-1)}v^{(2g-1)} + a_{(2g-3)}v^{(2g-3)} + \ldots + a_1 v + a_0)$;
3 $x := \dfrac{1+\varepsilon}{2} v + \dfrac{1-\varepsilon}{2} \left(\dfrac{w(-v+w)}{v+w} \right)$;
4 $y := -\varepsilon \sqrt{x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1 x + a_0}$;
5 **Retourner** (x,y).

Définition 3.3.17.

Dans la situation de l'algorithme 12, la fonction d'encodage de la courbe hyperelliptique $\mathbb{H}_g$ est la fonction $\psi_g : \mathcal{R}_g \to \mathbb{H}_g : r \mapsto \psi_g(r) = (x,y)$.

Théorème 3.3.18
L'algorithme 12 détermine un encodage déterministe presque-injectif $\psi_g : \mathcal{R}_g \to \mathbb{H}_g : r \mapsto \psi_g(r) = (x,y)$, en temps $\mathcal{O}(\log^{2+\circ(1)} q)$, où $Z_g = \mathbb{F}_q \backslash \mathcal{R}_g$ est un sous-ensemble de $\mathbb{F}_q$ d'au plus $(2g+1)^2 + 1$ éléments.

Preuve : Découle des remarques 3.3.14, 3.3.15 et des théorèmes 3.2.2, 3.2.5, 3.2.8, 3.2.11.

□

Lemme 3.3.19
Dans la situation du théorème 3.3.18 et de la définition 3.3.17, on a $\phi_g(r) = \phi_g(-r)$, $\forall r \in \mathcal{R}_g$ et $\#(\phi_g^{-1}(\phi_g(r))) = 2$, $\forall r \in \mathcal{R}_g$ (où $\#(f)$ désigne le cardinal de f) .

Preuve : Similaire à la preuve du **Lemma 1** dans [SBDK17].

□

Théorème 3.3.20
Dans la situation du théorème 3.3.18 et de la définition 3.3.17, nous avons les propriétés suivantes.

1. *Soit* (x, y) *un point de la courbe hyperelliptique* $\mathbb{H}_g$. *Alors* $(x, y) \in \mathrm{Im}(\psi_g)$ *si et seulement si* $uw(x + w)(-n_g - m_g s)$ *est un carré non nul dans* $\mathbb{F}_q$.
2. *Soit* $(x, y) \in \mathrm{Im}(\psi_g)$ *on définit* $\bar{r}$ *comme suit :* $\bar{r} = \sqrt{\frac{x+w}{uw(-n_g-m_g s)}}$ *si* $y \notin \sqrt{\mathbb{F}_q^2}$ *et* $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-n_g-m_g s)}}$ *si* $y \in \sqrt{\mathbb{F}_q^2}$. *Alors* $\bar{r} \in \mathcal{R}_g$ *et* $\psi_g(\bar{r}) = (x, y)$.

Preuve : Découle des remarques 3.3.15 et des théorèmes 3.2.3, 3.2.6, 3.2.9, 3.2.13.

□

3.4 Encodages presque-injectifs sur $\mathbb{H}_g$, $g \in \{6, 7, 8, 9\}$

En utilisant les résultats de la section précédente (**Section 3.3**), on vérifie que l'algorithme 12, la définition 3.3.17 et les théorèmes 3.3.18 et 3.3.20 peuvent être étendus aux familles de courbes hyperelliptiques $\mathbb{H}_g : y^2 = h_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1 x + a_0$ de genre $g \in \{6, 7, 8, 9\}$ où les coefficients de $\mathbb{H}_g$ sont donnés dans le tableau suivant.

Genre $\mathbf{g = 6}$	$a_0 = \frac{(s-78)}{78}w^{13}$, $a_1 = \frac{sw^{12}}{6}$, $a_3 = \frac{11}{3}sw^{10}$, $a_5 = \frac{33}{2}sw^8$, $a_7 = 22sw^6$, $a_9 = \frac{55}{6}sw^4$, $a_{11} = sw^2$
Genre $\mathbf{g = 7}$	$a_0 = \frac{(s-105)}{105}w^{15}$, $a_1 = \frac{sw^{14}}{7}$, $a_3 = \frac{13}{3}sw^{12}$, $a_5 = \frac{143}{5}sw^{10}$, $a_7 = \frac{429}{7}sw^8$, $a_9 = \frac{143}{3}sw^3$, $a_{11} = 13sw^4$, $a_{13} = sw^2$
Genre $\mathbf{g = 8}$	$a_0 = \frac{(s-136)}{136}w^{17}$, $a_1 = \frac{sw^{16}}{8}$, $a_3 = \frac{15}{3}sw^{14}$, $a_5 = \frac{91}{2}sw^{12}$, $a_7 = 143sw^{10}$, $a_9 = \frac{715}{4}sw^8$, $a_{11} = 91sw^8$, $a_{13} = \frac{35}{2}sw^4$, $a_{15} = sw^2$
Genre $\mathbf{g = 9}$	$a_0 = \frac{(s-171)}{171}w^{19}$, $a_1 = \frac{sw^{18}}{9}$, $a_3 = \frac{17}{3}sw^{16}$, $a_5 = 68sw^{14}$, $a_7 = \frac{884}{3}sw^{12}$, $a_9 = \frac{4862}{9}sw^{10}$, $a_{11} = 442sw^8$, $a_{13} = \frac{476}{3}sw^6$, $a_{15} = \frac{68}{3}sw^4$, $a_{17} = sw^2$

TABLE 3.4 – **Coefficients de** $\mathbb{H}_g$, $6 \leqslant g \leqslant 9$

3.5 Conclusion

Nous avons construit avec succès quatre encodages déterministes presque-injectifs $\phi_i, i = 1, 3, 4, 5$ sur quatre familles de courbes hyperelliptiques $\mathbb{H}_1, \mathbb{H}_3, \mathbb{H}_4$ et $\mathbb{H}_5$ de genre $g = 1, 3, 4, 5$ respectivement. Nous avons également construit un encodage déterministe unifié sur une courbe hyperelliptique de genre $g \leqslant 5$ qui généralise nos quatre encodages et ceux de Seck *et al.*. Dans chaque cas, nous avons montré dans quelles conditions on peut inverser la fonction d'encodage. Il serait intéressant, dans le futur, de voir si notre encodage en genre $g \leqslant 5$ peut être étendu à n'importe quel genre.

3.6 Implémentation de notre encodage unifié ψ_g (Section 3.3) avec SageMath [Dev17] disponible sur GitHub[Sec18]

```
class EncodingValidationOfParameters():
    def __init__(self, q, u, w, g = 2, s=None):
        self.R = FiniteField(q,"a")
        self.a = self.R.gen()
        self.poly = PolynomialRing(self.R,"x")
        self.x = self.poly.gen()
        self.q = q
        self.u = self.value2Fq(u)
        self.w = self.value2Fq(w)
        self.s = s
        self.g = g # g=genus of the curve
        self.alpha_g = 2^(2*g-1)-1
        self.beta_g = 4*g^2+2*g
        self.gamma_g = (2*g^2+g)^2
        self.mg = self.valueOfMg()
        self.ng = self.valueOfNg()
        #verification of the parameters
        self._chekingOfParameters()
        self._checkingValueOfTheParameterS()
        self.coefs = self.coefficients()
        self.f = self.function()
        self._chekingIfCurveIsHyperelliptic()

    def __repr__(self):
        eq = "y^2="+str(self.f)
        return "Hyperelliptic curve defined by {} over finite field".
           format(eq)+""\
            +" F_{}/< {} >".format(self.R.characteristic(),
                                   str(self.R.modulus()).replace("x","a"
                                      ))
    def function(self):
        g, R = self.g, self.R
        t0, tn = [R(self.coefs[0])], [self.x^(2*g+1)]
        poly_l = t0+[R(self.coefs[i])*self.x^(2*i-1) for i in range(1, g
           +1)]+tn
        return sum(poly_l)
    def valueOfMg(self):
        if self.g % 2 == 0: return (1/4)*self.alpha_g*self.beta_g
```

```
36          else : return (1/2)*self.alpha_g*self.beta_g
37      def valueOfNg(self):
38          if self.g % 2 == 0: return (1/2)*(2*self.g^2+self.g)^2
39          else : return (2*self.g^2+self.g)^2
40      def coefficients(self):
41          g, s, w, R = self.g, self.s, self.w, self.R
42          coefs = [(R(s-(2*g^2+g))/R(2*g^2+g))*w^(2*g+1),
43                   (s*w^(2*g))/R(g), R((2*g-1)/3)*s*w^(2*g-2)]
44          if g >= 3: coefs.append(s*w^2)
45          if g == 4: coefs.insert(3, R(7/2)*s*w^4)
46          if g == 5:
47              coefs.insert(3, R(42/5)*s*w^6)
48              coefs.insert(4, R(6)*s*w^4)
49          return coefs[:]
50      def value2Fq(self, value):
51          import re
52          search = re.compile("[a-zA-Z]+")
53          if isinstance(value, str):
54              return self.R(eval(search.sub("self.a",value).replace("^","
                    **")))
55          else: return self.R(value)
56      def _chekingOfParameters(self):
57          """ We check if the parameters  u,w,g,q satisfy the required
                conditions
58          """
59          p, g = self.R.characteristic(), self.g
60          if (p% 2 == 0) or ((2*g^2+g)%p == 0):
61              raise ValueError("The characteristic of F{}".format(self.q)+
                    ""\
62                   +" divise {}".format((2*g^2+g)))
63          if (self.q%8 != 7):
64              raise ValueError("q is different to 7 modulo 8".format(self.
                    q))
65          if self.w.is_zero() or self.u.is_zero(): raise ValueError("w = 0
                 or u = 0 in Fq ")
66          if self.u.is_square(): raise ValueError("u is square in Fq")
67          if self.g not in range(1,6): raise ValueError("The genus g out
                of range(1,5)")
68
69
70      def _checkingValueOfTheParameterS(self):
71          p, g = self.R.characteristic(), self.g
72          alpha_g, beta_g, gamma_g = self.alpha_g, self.beta_g, self.
                gamma_g
73          if self.s is None:
74              if alpha_g % p == 0: self.s = gamma_g/beta_g
75              else:
76                  delta_s = beta_g^2+4*alpha_g*gamma_g
77                  self.s = self.R((-beta_g+self.R(delta_s).square_root())
                        /(2*alpha_g))
78      def _chekingIfCurveIsHyperelliptic(self):
79          """ We check if f(x) doesn't have a double root"""
80          solution_f, solution_fprime = self.f.roots(), self.f.derivative
                ().roots()
81          f_roots = [root[0] for root in solution_f]
```

```
82          fprime_roots = [root[0] for root in solution_fprime]
83          if (set(f_roots) & set(fprime_roots)) != set():
84              raise ValueError("The function f={}".format(self.f)+" has
                    double roots")
85
86  class EncodingAndInvertGenusg(EncodingValidationOfParameters):
87      def __init__(self, q, u, w, g=2, s = None):
88          EncodingValidationOfParameters.__init__(self,q, u, w, g = g, s=s
                )
89      def _quadraticCharacter(self, value):
90          if self.R(value).is_zero(): return 0
91          elif self.R(value).is_square(): return 1
92          else: return -1
93      def valuesNotInDomainOfTheEncoding(self):
94          R, w, s, u, x = self.R, self.w, self.s, self.u, self.x
95          roots = self.f(w*(u*x^2*(R(-self.mg)*s+R(-self.ng))+R(-1))).
                roots()
96          return set([R(0)]) |  set([root[0] for root in roots if root
                !=[]])
97      def encode(self, value):
98          """ The encoding function psi(r)=(x,y)"""
99          R, w, s, u, poly, a = self.R, self.w, self.s, self.u, self.poly,
                 self.a
100         r = self.value2Fq(value)
101         if r in self.valuesNotInDomainOfTheEncoding():
102             raise ValueError("The encoding function is not defined at r
                    ={}".format(r))
103         v = w*(u*r^2*(R(-self.mg)*s+R(-self.ng))+R(-1))
104         e = epsilon = self._quadraticCharacter(self.f(v))
105         x = R((1+e)/2)*v+R((1-e)/2)*R(w*(-v+w)/(v+w))
106         y = R(-e)*R(self.f(x)).square_root()
107         return (x,y)
108     def decode(self, point):
109         """ The inverse of the encoding function"""
110         x,y,ng,mg = self.value2Fq(point[0]), self.value2Fq(point[1]),
                self.ng,self.mg
111         R, w, s, u, poly, a = self.R, self.w, self.s, self.u, self.poly,
                 self.a
112         if not (y^2-self.f(x)).is_zero():
113             raise ValueError("The given value is not a point of the
                    hyperelliptic curve")
114         if not (u*w*(x+w)*(R(-ng)+R(-mg)*s)).is_square():
115             raise Exception("u*w*(x+w)*(-ng-mg*s) is not a square in Fq"
                    )
116         r1 = ((x+w)/(u*w*(R(-ng)+R(-mg)*s))).square_root()
117         r2 = (R(2)*w/(u*(x+w)*(R(-ng)+R(-mg)*s))).square_root()
118         return "The preimages of ({} , {}) by the encoding function".
                format(x,y)+""\
119                 +" are equal to ({} , {}) or ({} , {})".format(r1, R(-r1
                    ),r2, R(-r2))
120
121 #First example
122 q = 2^521-1
123 fe = EncodingAndInvertGenusg(q=q,u=3,w=5,g=2); print(fe)
124 pt = fe.encode(121); print("\n (x,y) = "+str(pt)+"\n")
```

```
125 dc = fe.decode(pt); print("\n"+str(dc)+"\n")
126 # Second example : we change the genus of the curve
127 ge = EncodingAndInvertGenusg(q=q,u=3,w=5,g=3); print(ge)
128 pt2 = ge.encode(121); print("\n (x,y) = "+str(pt2)+"\n")
129 dc2 = ge.decode(pt2); print("\n"+str(dc2)+"\n")
130 \end{verbatim}
```

Chapitre 4

Une note sur l'encodage et le hachage sur les courbes elliptiques et hyperelliptiques [1]

Contents

1. Ce chapitre est basé sur l'article publié suivant :

Diarra N., Seck M. and Sow D.. A Note on Encoding and Hashing into Elliptic and Hyperelliptic Curves . In A Collection of Papers in Mathematics and Related Sciences, a festschrift in honour of the late Galaye Dia (Editors : Seydi H., Lo G.S. and Diakhaby A.). Spas Editions, Euclid Series Book, pp. 565 –593. (2018) Doi : 10.16929/sbs/2018.100-07-01

4. Une note sur l'encodage et le hachage sur les courbes elliptiques et hyperelliptiques[2]

Nous présentons l'état de l'art des encodages et des fonctions de hachage existants sur les courbes (hyper)elliptiques. Nous proposons également une fonction de hachage indifférentiable sur la Jacobienne de certaines familles de courbes hyperelliptiques de genre $g \leqslant 9$ en utilisant les formules unifiées de Seck et Diarra (AFRICACRYPT-2018).

4.1 Introduction

Pour hacher sur la Jacobienne d'une courbe (hyper)elliptique, nous avons besoin d'une fonction qui associe de manière déterministe un élément d'un corps fini $\mathbb{F}_q$ à un point de la courbe. Une telle fonction est appelée un encodage. Nous avons besoin de fonctions d'encodage sur des Courbes (Hyper)Elliptiques pour globalement deux raisons :

1. **représentation de points** : si un encodage f est presque-injectif (il est injectif quand il est restreint à un certain sous-ensemble de $\mathbb{F}_q$, voir [BHKL13]) et inversible (on peut trouver une antécédent pour n'importe quel élément de l'ensemble des images), il peut être utilisé pour représenter chaque point de l'ensemble des images par une chaîne de bits uniforme aléatoire, comme ce que Bernstein et al. ont fait dans Elligator(1 et 2) [BHKL13].
2. **hachage indifférentiable** : si un encodage f est bien distribué (c'est-à-dire que la somme de ses images relativement à un caractère associé fixé, dépendant du paramètre de sécurité, est bornée), on peut utiliser f pour concevoir une fonction de hachage indifférentiable sur la courbe cible, en suivant le procédé général proposé dans [FFS+13, BCI+10].

Le problème d'encodage sur les courbes elliptiques et hyperelliptiques a été étudié par de nombreux auteurs. Ce problème peut être formulé comme suit : Étant donnés un élément r dans un corps fini $\mathbb{F}_q$ et une courbe (hyper)elliptique H, associer cet élément r à un élément de H. La première proposition (à notre connaissance) de fonctions d'encodage sur des courbes elliptiques a été faite par Boneh et Franklin en 2001 pour leur schéma IBE [BF01]. Depuis lors, il y a eu de nombreuses propositions d'encodage sur différentes formes de courbes elliptiques :

- pour le modèle de Weierstrass $\mathbf{E} : \mathbf{y^2 = x^3 + a_2x^2 + a_4x + a_6}$: Boneh et Franklin (2001) [BF01], Shallue et van de Woestjine (2006) [SvdW06], Icart (2009) [Ica09], Brier *et al.* (2010) [BCI+10], Fouque et Tibouchi (2012) [FT12], Fouque *et al.* (2013) [FJT13], Bernstein *et al.*(2013) [BHKL13], Diarra *et al.* (2017) [DDK17], Seck et Diarra (2018)[SD18] ;
- pour la forme Hessienne $\mathbf{E} : \mathbf{x^3 + y^3 + 1 = 3dxy}$: Kammerer *et al.* (2010) [KLR10], Farashahi (2011) [Far11] ;
- pour la forme de Montgomery $\mathbf{E_{a,b}} : \mathbf{by^2 = x^3 + ax^2 + x}$: Yu *et al.*(2013)[YWLT13] ;
- pour le modèle d'Edwards $\mathbf{E_d} : \mathbf{x^2 + y^2 = 1 + dx^2y^2}$: Bernstein *et al.* (2013) [BHKL13], He *et al.* (2016)[YWL+16], Diarra *et al.* (2017) [DDK17] ;
- pour le modèle de la quartique de Jacobi (Jacobi Quartic) $\mathbf{E} : \mathbf{y^2 = x^4 + 2bx^2 + 1}$: Yu *et al.* (2015)[YWL+15] ;
- pour les modèles de Huff $\mathbf{E} : \mathbf{ax(y^2 - c) = by(x^2 - d)}$ et $\mathbf{E_{a,b}x(ay^2 - 1) = y(bx^2 - 1)}$: He *et al.* (2016) [YWL+16] and Diarra *et al.* (2017) [DDK17] ;

- pour le modèle d'intersection de Jacobi (Jacobi Intersection model) $\mathbf{au^2 + v^2 = 1}$, $\mathbf{bv^2 + w^2 = 1}$: He *et al.* (2018) [HYW17].

Pour le cas des courbes hyperelliptiques également, de nombreux auteurs ont proposé des encodages pour différentes familles de courbes, comme les encodages d'Ulas (2007) [Ula07], Kammerer *et al.* (2010) [KLR10], Seck *et al.* (2017 et 2018) [SBDK17, SD18].

En utilisant les encodages existants, des constructions de fonctions de hachage dans la Jacobienne de certaines familles de courbes (hyper)elliptiques ont été proposées. La première construction était la méthode *try-and-increment* due à Boneh *et al.* (2001) [BLS01]). D'autres constructions ont été proposées par Brier *et al.* (2010) [BCI+10] : constructions de la forme $H = f \circ h$, $H = f \circ h_1 + h_2\mathbb{G}$ et $H = f \circ h_1 + f \circ h_2$ généralisées ultérieurement par Farashahi et al. (2013) [FFS+13] ($H = f \circ h_1 + f \circ h_2 + \ldots + f \circ h_s$).

Nos contributions :

— Nous donnons d'abord l'état de l'art de tous les encodages existants (à notre connaissance) sur les courbes elliptiques et hyperelliptiques ainsi que les différentes techniques de hachage sur la Jacobienne d'une courbe (hyper)elliptique.

— Ensuite, nous faisons une analyse comparative des différentes méthodes de construction d'encodages déterministes. Nous comparons également les méthodes existantes de construction de fonctions de hachage indifférentiables.

— A la fin, nous proposons de nouvelles formules unifiées pour le hachage indifférentiable sur la jacobienne de certaines familles de courbes hyperelliptiques (de genre $g \leqslant 9$).

Organisation du chapitre : Ce chapitre est organisé comme suit.

— Dans la section 4.2, nous faisons l'état de l'art de tous les encodages sur les courbes (hyper)elliptiques.

— Dans la section 4.3, nous faisons l'état de l'art des méthodes de hachage sur la Jacobienne de courbes elliptiques et hyperelliptiques.

— Dans la section 4.4, nous proposons une nouvelle fonction de hachage indifférenciable utilisant l'encodage unifié de Seck et Diarra à AFRICACRYPT-2018 [SD18]. Et nous concluons à la Section 4.5.

4.2 Etat de l'art des encodages sur les courbes (hyper)elliptiques

4.2.1 Encodages sur les courbes elliptiques

Nous donnons dans cette sous-section, les principaux encodages existants sur différentes formes de courbes elliptiques telles que la forme Weierstrass, la forme Montgomery, la forme quartique de Jacobi, la forme de Huff, la forme d'Edwards etc.

Encodage sur la forme de Weierstrass

• **Méthode naïve :** : Pour une courbe elliptique $E_{a,b}$: $y^2 = x^3 + ax + b$, la façon la plus simple de construire un point de $E_{a,b}$ est d'utiliser la méthode naïve. L'idée est de choisir une coordonnée x et d'essayer de déduire la coordonnée y en calculant une racine carrée : on choisit un élément aléatoire $u \in \mathbb{F}_q^*$ et on calcule $u^3 + au + b$; puis on teste si $u^3 + au + b$ est un carré dans $\mathbb{F}_q$. Si oui, alors on retourne $(x, y) = (u, \pm\sqrt{u^3 + au + b})$ comme point de la courbe. Sinon, on peut choisir un autre u dans $\mathbb{F}_q$ et réessayer. Mais cette méthode a au moins un inconvénient, c'est-à-dire qu'elle ne peut pas fonctionner en temps constant : le nombre d'opérations dépend de l'entrée u. En pratique, l'entrée u est le message m que l'on veut hacher ; ainsi, l'exécution de cet algorithme peut permettre à l'attaquant de deviner certaines informations sur m.

• **Encodage sur des courbes supersingulières** : Une courbe elliptique E sur $\mathbb{F}_q$ telle que $|E(\mathbb{F}_q)| = q + 1$ est une courbe *supersingulière*. Pour $q \equiv 2 \bmod 3$, la courbe définie par $E_b : y^2 = x^3 + b$ est une courbe supersingulière. Dans leur schéma basé sur l'identité[BF01], Boneh et Franklin ont proposé la fonction $f : u \mapsto ((u^2 - b)^{\frac{1}{3}}, u)$ qui construit un point de E_b, étant donné un élément quelconque $u \in \mathbb{F}_q$. Cette fonction leur permet de construire la clef publique Q_{id}(un point sur la courbe supersingulière) correspondant à l'identité id $\in \{0,1\}^*$. Mais il est bien connu que les courbes supersingulières sont inutiles pour les préoccupations cryptographiques (utilisant le DLP) à cause de l'attaque MOV [MOV93] : c'est à dire que le DLP sur E_b peut être réduit au DLP dans $\mathbb{F}_q$. Pour éviter ces attaques, il faut utiliser un grand q.

• **Algorithme de SWU** *(2006)* : Dans [SvdW06], Shallue et van de Woestjine ont proposé un algorithme qui génère, en temps polynomial, un point d'une courbe elliptique E sur $\mathbb{F}_q$ (de caractéristique $\geqslant 5$) donnée par son équation de Weierstrass $y^2 = g(x) = x^3 + a_2x^2 + a_4x + a_6$, avec $a \neq 0$. Leur encodage est basé sur l'égalité de Skalba[Ska05] : il existe quatre fonctions rationnelles non constantes $X_1(t), X_2(t), X_3(t), X_4(t)$ telles que :

$$g(X_1(t)) \cdot g(X_2(t)) \cdot g(X_3(t)) = (X_4(t))^2$$

Il s'en suit qu'au moins un des $g(X_i(u))$ doit être un résidu quadratique dans le corps fini $\mathbb{F}_q$, étant donné un élément fixé $u \in \mathbb{F}_q$. On peut alors déduire une fonction d'encodage sur la courbe E : $y^2 = g(x)$. Il suffit de poser $(x, y) = (X_i(u), \sqrt{g(X_i(u))})$, où i est le plus petit indice tel que $g(X_i(u))$ soit un résidu quadratique.
Même si cette construction définit un encodage à temps constant, elle présente au moins un inconvénient. En effet, les fonctions rationnelles $X_i(t)$ sont complexes et difficiles à implémenter. Et il n'y a pas d'algorithme déterministe s'exécutant en temps polynomial pour calculer une racine carrée dans $\mathbb{F}_q$, à moins de faire des hypothèses supplémentaires sur q. Par exemple, quand $q \equiv 3 \bmod 4$, alors le calcul d'une racine carrée est simplement une exponentiation.
Notez que certains auteurs ont utilisé cette méthode générale pour construire des encodages pour des familles particulières de courbes elliptiques (comme les courbes BN [4.2.1], les courbes de Huff généralisées [2], etc.) et les courbes hyperelliptiques ([4.2.2]).

• **Encodage d'Icart** (*2009*) : Soit $q = 2 \bmod 3$; donc la fonction $x \mapsto x^3$ est une bijection et ensuite le calcul d'une racine cubique peut être fait comme une exponentiation. Dans [Ica09], Icart a défini une nouvelle fonction d'encodage basée sur l'idée suivante : intersecter la droite $y = ux + v$ avec la courbe de Weierstrass $E_{a,b}$: $y^2 = x^3 + ax + b$, avec $a, b \in \mathbb{F}_q$. Il a défini la fonction d'encodage : $f_{a,b}$: $\mathbb{F}_q \to E_{a,b} : u \mapsto f_{a,b}(u) = (x, ux + v)$ où $x = (v^2 - b - \frac{u^6}{27})^{1/3} + \frac{u^2}{3}$ et $v = (3a - u^4)/6u$. Comme montré dans son article, cette fonction présente de nombreuses propriétés intéressantes. En fait, elle peut être implémentée en temps polynomial avec des opérations $O(\log^3 q)$. La fonction inverse $f_{a,b}^{-1}$ est aussi calculable en temps polynomial. Icart a aussi montré que $|f_{a,b}^{-1}(P)| \leqslant 4$, pour un point P sur la courbe elliptique. Ceci résulte du fait que pour calculer $f_{a,b}^{-1}(P)$, il suffit de résoudre le polynôme de degré 4 sur $\mathbb{F}_q$:

$$u^4 - 6u^2x + 6uy - 3a = 0$$

De plus, Icart a montré que le cardinal de l'ensemble des images $\mathrm{Im}(f_{a,b})$ est supérieur à $q/4$ et a fait la conjecture suivante (prouvée plus tard par Tibouchi et Fouque dans [FJT13]) : la taille de $\mathrm{Im}(f_{a,b})$ est approximativement égale à $\frac{5}{8}$ de la taille de la courbe $E_{a,b}$.

• **Encodage simplifié d'Ulas** (*2010*) : Brier *et al.* [BCI+10] ont proposé une version simplifiée de l'encodage d'Ulas quand $q \equiv 3 \bmod 4$ (Ulas a généralisé l'algorithme SWU aux courbes hyperelliptiques de la forme $y^2 = x^n + ax + b$ ou $y^2 = x^n + ax^2 + bx$). En fait, les auteurs ont obtenu des formules pour la forme de Weierstrass $y^2 = x^3 + ax + b$, en explicitant les fonctions rationnelles décrites dans l'algorithme SWU. Leurs fonctions sont définies par la proposition suivante :

Proposition 4.2.1
Soit $q \equiv 2 \bmod 3$, *et* $g(x) = x^3 + ax + b$ *(avec* $a, b \in \mathbb{F}_q^*$*). Soit* $X_2(t) = -\frac{b}{a}\left(1 + \frac{1}{t^4 - t^2}\right)$, $X_3(t) = -t^2 X_2(t)$, $U(t) = t^3 g(X_2(t))$ *Alors* $U(t)^2 = -g(X_2(t)) \cdot g(X_3(t))$.

Cela leur permet de définir un encodage $\alpha-$weak (avec $\alpha = 8N/q$ et N l'ordre de la courbe) dans le modèle de Weierstrass $y^2 = x^3 + ax + b$.

• **Encodage sur les courbes BN** (*2012*) : Une courbe de Barreto-Nahrig (BN) [FT12] est une courbe elliptique "pairing-friendly" de la forme E_b : $y^2 = g(x) = x^3 + b$ sur un corps fini $\mathbb{F}_q$, où $q \equiv 1 \bmod 3$, $\#E_q(\mathbb{F}_q)$ est premier et b est le plus petit entier > 0 tel que $1 + b$ soit un carré non nul dans $\mathbb{F}_q$. Une courbe BN est alors un cas particulier de courbes prises en compte par l'algorithme de SWU, avec $a_2 = a_4 = 0$ et $a_6 = b$.
A partir de ce fait, Fouque et Tibouchi [FT12] ont proposé un encodage pour les courbes BN, en rendant explicite la fonction d'encodage de Shallue et Woestjine (voir paragraphe précédent). Cela conduit à la fonction d'encodage suivante :

$$f : \quad \mathbb{F}_q \to E_b(\mathbb{F}_q),\ t \mapsto f(t) = \begin{cases} \left(\frac{-1+\sqrt{-3}}{2}, \sqrt{1+b}\right) & \text{si } t = 0 \\ (x_i, \chi(t) \cdot \sqrt{g(x_i)}) & \text{si } t \neq 0 \end{cases}$$

où χ est le caractère quadratique sur $\mathbb{F}_q$ et i est le plus petit indice dans $\{1, 2, 3\}$ tel que $g(x_i)$ soit un carré (les valeurs de x_1, x_2, x_3 sont données explicitement dans le même

papier [FT12]). Les auteurs ont également donné une estimation de la taille de l'ensemble des images (environ 9/16 de la taille de la courbe E_b).

• **Encodages injectifs** (*2013*) : Le problème de la construction d'encodages injectifs pour une grande famille de courbes elliptiques (incluant toutes les courbes avec un point d'ordre 4 et seulement un point d'ordre 2, et les courbes avec des points d'ordre 2) a été étudié dans [FJT13] par Fouque *et al.* Dans leur article, ils ont proposé une façon de construire un encodage injectif pour une courbe elliptique vue comme un quotient d'une courbe hyperelliptique impaire (par exemple une courbe hyperelliptique de la forme $y^2 = f(x) = x^{2g+1} + a_{2g}x^{2g} + \ldots + a_1x$). Les courbes elliptiques compatibles avec leurs encodages sont de la forme : $y^2 = x^3 \pm 4x^2 + Ax$ (voir [FJT13], pages 11-12). Les auteurs ont proposé plus tard une version revisitée [FJT13] dans laquelle ils ont donné des formules pour calculer l'encodage injectif pour le modèle court de Weierstrass (et aussi pour la forme d'Edwards en utilisant une équivalence birationnelle). Ils encodent à partir d'un sous-ensemble I de $\mathbb{F}_q$ de cardinal $(q+1)/2$ et tel que $I \cap -I = \{0\}$.

Certains auteurs ont ensuite utilisé leur méthode générique pour construire des encodages pour des familles particulières de courbes elliptiques (comme les courbes d'Edwards par Bernstein *et al.* [BHKL13], ou les courbes de Huff généralisées par Diarra *et al.* [DDK17]).

• **Elligator 2** (*2013*) : Dans le même article [BHKL13] que Elligator 1 (pour les courbes d'Edwards), Bernstein *et al.* ont introduit une seconde fonction d'encodage qu'ils ont appelée *Elligator 2*, pour le modèle de Weierstrass $E_{A,B}$: $y^2 = x^3 + Ax^2 + Bx$ avec $AB(A^2 - 4B) \neq 0$ sur tout corps fini $\mathbb{F}_q$ de caractéristique impaire. Elligator 2 associe n'importe quel élément d'un sous-ensemble particulier R de $\mathbb{F}_q$ à un point de la courbe de Weierstrass. Et, pour un choix approprié de q (à savoir $q \equiv 1 \bmod 4$), on obtient $R = \mathbb{F}_q$, par exemple l'encodage couvre tous les éléments de $\mathbb{F}_q$. Dans Elligator 2, pour encoder un élément non nul $r \in R = \{r \in \mathbb{F}_q : \ 1 + ur^2 \neq 0, \ A^2ur^2 \neq B(1+ur^2)^2\}$, on doit procéder comme suit : d'abord on calcule $v = -A(1+ur^2)$, puis on retourne (x, y) comme point sur la courbe $E_{A,B}$ où $x = v$, $y = -\sqrt{x^3 + Ax^2 + Bx}$, si $v^3 + Av^2 + Bv$ est un résidu quadratique dans $\mathbb{F}_q$; et $x = -v - A$, $y = \sqrt{x^3 + Ax^2 + Bx}$ dans le cas contraire.

Elligator 2 est presque-injectif ; il peut donc être utilisé pour représenter des points sur la courbe comme des chaînes de bits aléatoires uniformes, comme le montrent les auteurs.

• **Encodage pour le modèle de Weierstrass en caractéristique deux** : Une courbe elliptique sur un corps binaire $\mathbb{F}_{2^n}$ est une courbe de la forme $E_{a,b}$: $y^2 + xy = x^3 + ax^2 + b$, où $a, b \in \mathbb{F}_{2^n}$. Icart a appliqué sa méthode [Ica09] pour le modèle court de Weierstrass (avec le calcul d'une racine cubique) pour obtenir un encodage simple et efficace pour les courbes elliptiques binaires (en supposant que n est impair, et donc que $x \mapsto x^3$ est une bijection). Cet encodage en caractéristique deux satisfait les mêmes propriétés qu'en caractéristique impaire.
Brier *et al.* [BCI+10] ont également proposé une variante de leur encodage sur la forme de Weierstrass sur des corps binaires, en utilisant l'algorithme de Shalue-Woestjine.

• **Encodage pour le modèle Weierstrass en caractéristique trois** : Les courbes elliptiques ordinaires sur $\mathbb{F}_3^n$ sont représentées par une équation de Weierstrass $E_{a,c}$: $y^2 =$

$f(x) = x^3 + ax^2 + c$, avec le discriminant $\Delta = -a^3b \neq 0$. Brier *et al.* [BCI+10] ont été les premiers à proposer un encodage déterministe pour les courbes elliptiques en caractéristique trois. En utilisant la méthode Elligator 2, Diarra *et al.* ont aussi proposé [DDK17] un encodage direct de $\mathbb{F}_3^n$ sur de telles courbes (où $-ac$ est un carré). Étant donné un élément non nul r dans $\mathbb{F}_3^n$, leur encodage fonctionne comme suit :

— on pose $s^2 = -c/a$ ($-ac$ est un carré) et on calcule $v = \dfrac{ur^2 - s^2}{s}$;
— on retourne (x, y) comme un point sur $E_{a,c}$, où

$$\begin{cases} x = v \text{ et } y = -\sqrt{f(x)} & \text{si } f(v) \text{ est un carré ;} \\ x = \dfrac{sv}{s+v} \text{ et } y = \sqrt{f(x)} & \text{sinon.} \end{cases}$$

Cet encodage partage toutes les propriétés d'Elligator 2, comme la presque-injectivité et la caractérisation de l'ensemble image.

Encodage sur les courbes Hessiennes

• **Kammerer *et al.*** *(2010)* : [KLR10] ont proposé des encodages pour divers modèles de courbes (hyper)elliptiques, dont le modèle Hessien E_d : $x^3 + y^3 + 1 = 3dxy$ ($d \neq 1$) sur $\mathbb{F}_q$, avec $q \equiv 2 \bmod 3$. Pour définir cet encodage, ils ont résolu des équations de courbe en radicaux (comme dans la méthode d'Icart [Ica09]) pour le modèle de Weierstrass E_a : $Y^2 + XY + aY = X^3$, ce qui est birationnellement équivalent à la courbe E_d. Leur encodage est un week-encodage dans le sens de la définition (2.2.6), et est de 2 : 1 (comparé à l'encodage d'Icart qui est de 4 : 1). Leur encodage peut être étendu à $\mathbb{F}_q$ et peut être défini quand $d = 2$ (voir [KLR10]).

• **Farashahi** *(2011)* : Soit $d \in \mathbb{F}_q$ avec $d^3 \neq 1$. Une **courbe Hessienne** (une courbe avec un point d'ordre 3) H_d sur $\mathbb{F}_q$ est donnée par l'équation $x^3 + y^3 + 1 = 3dxy$. Pour $q \equiv 2 \bmod 3$, Farashahi [Far11] a appliqué la méthode d'Icart à l'ensemble $H_d(\mathbb{F}_q)$ des points $\mathbb{F}_q$-rationnels de H_d. Il a obtenu la fonction : $h_d : \mathbb{F}_q \to H_d(\mathbb{F}_q)$ défini par $h_d(u) =$

— (x, y) si $u \neq -1$, où $x = -u\left(\dfrac{d^3u^3+}{u^3+1}\right)^{1/3}$, $v = -\left(\dfrac{d^3u^3+}{u^3+1}\right)^{1/3} + du$;
— $\mathcal{O}$ si $u = -1$, où $\mathcal{O}$ est le point à l'infini.

La fonction h_d est bien définie puisque $h_d(u)$ est un point de $H_d(\mathbb{F}_q)$, pour $u \in \mathbb{F}_q$. Cette fonction d'encodage pour les courbes Hessiennes est moins générale que celle d'Icart, mais elle a beaucoup d'autres propriétés intéressantes. En fait, Farashahi a montré que la taille de l'espace image $h_d(\mathbb{F}_q)$ est d'au moins $q/2$ et que la fonction inverse h_d^{-1} peut être facilement déterminer. Il a aussi étudié la possibilité d'extraire des bits aléatoires de l'image $h_d(u)$ d'un élément $u \in \mathbb{F}_q$.

Encodage sur les courbes d'Edwards

• **Elligator 1 : Un encodage injectif pour les courbes d'Edwards** *(2013)* : Dans [BHKL13], Bernstein *et al.* ont défini un encodage qui fait correspondre un élément

de $\mathbb{F}_q$ à un point de la courbe d'Edwards $E_d: x^2+y^2 = 1+dx^2y^2$, avec $d \notin \{0,1\}$ et d n'est pas un carré. Leurs travaux utilisent la méthode générale proposée par Fouque et Tibouchi dans [FT10]. Notez que, dans leur article mis à jour [FJT13], Fouque *et al.* ont aussi proposé un encodage explicite pour les courbes d'Edwards, basé sur leurs travaux précédents. La méthode d'encodage de Bernstein *et al.* dans [BHKL13] est décrite comme suit : Soit q une puissance d'un nombre premier congruent à 3 modulo 4. Soit s un élément non nul de $\mathbb{F}_q$ avec $(s^2-2)(s^2+2) \neq 0$. On pose $c = 2/s^2$, alors $c(c-1)(c+1) \neq 0$. On définit $r = c + 1/c$ et $d = -(c+1)^2/(c-1)^2$. Alors $r \neq 0$, et d n'est pas un carré. Les éléments suivants de $\mathbb{F}_q$ sont définis pour chaque $t \in \mathbb{F}_q \backslash \{0,1\}$:

$$u = (1-t)/(1+t), \qquad v = u^5 + (r^2-2)u^3 + u;$$
$$X = \chi(v)u, \qquad Y = (\chi(v)v)^{(q+1)/4}\chi(v)\chi(u^2+1/c^2);$$
$$x = (c-1)sX(1+X)/Y, \qquad y = (rX-(1+X)^2)/(rX+(1+X)^2).$$

De plus $x^2+y^2 = 1+dx^2y^2$; $uvXYx(y+1) \neq 0$ et $Y^2 = X^5 + (r^2-2)X^3 + X$. Ainsi, cet algorithme décrit un encodage de $\mathbb{F}_q$ à E_d. Les auteurs ont aussi montré que l'ensemble image de leur encodage peut être facilement décrit explicitement. De plus, il devient un encodage injectif dans E_d si l'on considère seulement la moitié des points de $\mathbb{F}_q$.

• **AIIE-For-Edwards** (*2017*) : Diarra *et al.* ont proposé dans [DDK17] un algorithme pour encoder directement un élément de $\mathbb{F}_q$ (avec $\text{char}(\mathbb{F}_q) \neq 2$) à un point d'une courbe d'Edwards $E_d: x^2+y^2 = 1+dx^2y^2$ ($d \neq \pm 1, -2$ et d n'est pas un carré), en utilisant la méthode d'Elligator-2 [BHKL13] de Bernstein *et al.* (et sans aucune équivalence birationnelle). Pour encoder en E_d, on doit d'abord choisir un élément u qui n'est pas un carré dans $\mathbb{F}_q$, et définir l'ensemble $\mathcal{R} = \{r \in \mathbb{F}_q^* : ur^2+1 \neq 0,\ ur^2(1-d)-(1+3d) \neq 0,\ ur^2(1+3d)-(1-d) \neq 0\}$. L'encodage est alors donné par l'algorithme suivant :

Algorithme 13 : AIIE-For-Edwards

Entrée : $r \in \mathcal{R}$;
Sortie : Un point $(x,y) \in E_d: x^2+y^2 = 1+dx^2y^2$;

1. $v = \dfrac{(d-1)ur^2-3-d}{ur^2(d-1)+1+3d} \qquad \varepsilon = \chi[(1-v^2)(1-dv^2)]$
2. $x = \dfrac{\varepsilon(v+1)(dv+1)+dv^2-1}{2d(v+1)+(1-d)} \qquad y = -\varepsilon\sqrt{\dfrac{1-x^2}{1-dx^2}}$
3. **Retourner** $(x,y) \in E_d$.

Les auteurs ont montré que cet encodage est presque-injectif et inversible. Ils ont aussi donné quelques exemples de Courbes (SafeCurves) [BL14] qui sont appropriées pour cet encodage.

Encodage sur les courbes de Huff

• **Encodage sur des courbes de Huff généralisées : Brief SWU encoding** (*2015*) : Dans [HYW15], He *et al.* ont proposé deux méthodes pour encoder sur des courbes de Huff généralisées données par $E: ax(y^2-c) = by(x^2-d)$, avec $abcd(a^2c-b^2d) \neq 0$.

La première, que nous décrivons ici, est appelée encodage Brief SWU encoding. Cet encodage est une fonction de la forme $f = \psi \circ \phi$, où ϕ est un encodage de $\mathbb{F}_q$ à la courbe de Weierstrass $E' : \ y^2 = g(x) = x^3 + (b^2d + a^2c)x^2 + (a^2b^2cd)x$. Comme mentionné par les auteurs, l'encodage ϕ est une adaptation de l'encodage d'Ulas (basé sur la méthode SWU [4.2.1]) pour les courbes elliptiques définies par $y^2 = x^n + Ax^2 + Bx$ sur $\mathbb{F}_q$, avec $q \equiv 3 \bmod 4$. Cela fonctionne comme suit, étant donné $u \in \mathbb{F}_q$:

— si $u = 0$, retourner $(0, 0)$;

— sinon, calculer $X(u) = \dfrac{a^2b^2cd(u^2-1)}{a^2c + b^2d}$ et $g(X(u))$;

— calculer $Y(u) = -\dfrac{a^2b^2cd}{a^2c + b^2d}\left(1 - \dfrac{1}{u^2}\right)$ et $g(Y(u))$;

— retourner $(s, t) = (X(u), -\sqrt{g(X(u))})$ ou $(s, t) = (Y(u), \sqrt{g(Y(u))})$ (d'après l'égalité du Skalba, il s'ensuit que soit $g(X(u))$ soit $g(Y(u))$ est un résidu quadratique dans $\mathbb{F}_q$).

La fonction ψ définit l'équivalence birationnelle de la forme de Weierstrass ci-dessus avec la courbe de Huff généralisée :

$$\psi : \ E'(\mathbb{F}_q) \to E(\mathbb{F}_q), \ (s, t) \mapsto (x, y) = \left(\frac{bd(s + a^2c)}{t}, \frac{ac(s + b^2d)}{t}\right)$$

L'encodage f couvre tout $\mathbb{F}_q$ et est bien distribué (au sens de [FFS+13]). Les auteurs ont aussi donné une estimation de la taille de l'ensemble des images, en utilisant le théorème de densité de Tchebotarev (voir [HYW15]) comme l'ont fait Fouque et Tibouchi pour estimer la taille de l'encodage d'Icart.

• **Encodage sur des courbes de Huff généralisées : Cube Root Encoding** (*2015*) : Le second encodage pour les courbes de Huff généralisées proposé par He *et al* dans [HYW15] est le Cube root Encoding. Comme son nom l'indique, cet encodage est basé sur le calcul des racines cubiques dans $\mathbb{F}_q$, où $q \equiv 2 \bmod 3$. Cette condition sur q leur permet de calculer efficacement les racines cubiques, par $a^{1/3} = a^{(2q-1)/3}$. L'encodage fonctionne comme suit, étant donné $u \in \mathbb{F}_q$:

— Calculer $t = u^2 - a^2c - b^2d$;

— Puis calculer $r = \dfrac{1}{2}(a^2b^2cd - \dfrac{1}{3}t^2)$.

Cet encodage est un weak-encodage (au sens de Brier et al[BCI+10]), qui permettra de construire une fonction de hachage indifférentiable sur cette courbe de Huff.

• **AIIE-For-Gen-Huff : Encodage sur des courbes de Huff généralisées** (*2017*) : Suivant l'idée de Fouque *et al.* dans [FJT13] pour la construction d'encodages injectifs, Diarra *et al.* ont proposé un encodage explicite pour le modèle de Huff généralisé $E_{a,b} : \ x(ay^2 - 1) = y(bx^2 - 1)$ sur $\mathbb{F}_q$, avec $q \equiv 3 \bmod 4$ et $ab(a - b) \neq 0$ (notons que Bernstein *et al.* ont fait une construction identique pour le modèle d'Edwards [BHKL13]). Nous résumons ici leur encodage. Pour encoder les éléments de $\mathbb{F}_q$ sur un point de $E_{a,b}$, il

faut choisir un élément $c \in \mathbb{F}_q$ tel que $c(c-1)(c+1) \neq 0$, et calculer $A = -(c - \frac{1}{c})^2$. Soit $s = \dfrac{c+1}{c-1}$ et $\lambda \in \mathbb{F}_q^*$; On pose $b = \dfrac{\lambda^2}{1-s^2}$ et $a = -bs^2$. Ensuite l'algorithme suivant montre comment encoder n'importe quel $z \in \mathbb{F}_q \backslash \{-1, 1\}$ en un point de $E_{a,b} \backslash (0,0)$ (et $1, -1$ sont envoyés sur le point $(0,0)$).

Algorithme 14 : AIIE-For-Gen-Huff

Entrée : $z \in \mathbb{F}_q \backslash \{-1, 1\}$,

Sortie : Un point $(x, y) \in \mathbf{E_{a,b}} : \mathbf{x(ay^2 - 1) = y(bx^2 - 1)}$;

1. $u = (1-z)/(1+z), \qquad v = -u^5 + (c^2 + 1/c^2)u^3 - u$
2. $X = \chi(v)u, \qquad Y = \chi(v \cdot (c^2u^2 - 1))\sqrt{\chi(v)v}$
3. $\alpha = \dfrac{\lambda}{2} = \dfrac{\sqrt{a+b}}{2}, \quad x = \frac{(1+X)(bX - \alpha^2(1+X)^2)}{b\alpha Y}, \quad y = \frac{(1+X)(aX - \alpha^2(1+X)^2)}{a\alpha Y}$
4. **Retourner** $(x, y) \in E_{a,b}$

Cet algorithme, en plus d'encoder sur la courbe de Huff, permet également d'encoder sur la courbe Hyperelliptique $Y^2 = -X^5 + (2 - A)X^3 - X$ (on peut vérifier que (X, Y) est bien un point de cette courbe).

• **AIIE-For-Class-Huff : Encodage sur les courbes de Huff classiques** (*2017*) : Diarra *et al.* [DDK17] ont également proposé un encodage pour les courbes de Huff classiques, qui sont des courbes de la forme $\alpha x(y^2-1) = \beta y(x^2-1)$ avec $\alpha\beta(\alpha^2-\beta^2)(\alpha^2+\beta^2) \neq 0$. On peut remarquer que ces courbes sont une classe particulière du modèle généralisé $x(ay^2 - 1) = y(bx^2 - 1)$ (il suffit de poser $a = \alpha^2$, $b = \beta^2$ et d'utiliser le changement des variables $(x, y) \to (x' = \beta x, y' = \alpha y)$). Mais, comme les auteurs l'ont remarqué, leur méthode d'encodage dans le modèle généralisé ne pouvait pas fonctionner directement pour le modèle classique (puisque le produit ab doit être un résidu non quadratique dans $\mathbb{F}_q$). Ils ont donc suggéré une autre méthode, qui est basée sur Elligator-2 de Bernstein *et al.*, pour encoder directement (c'est-à-dire sans aucune équivalence birationnelle) dans le modèle classique. Rappelons qu'Elligator-2 définit un encodage injectif pour les courbes de Weierstrass de la forme $y^2 = x^3 + Ax^2 + Bx$. Avec cette méthode, pour encoder sur un autre modèle de courbes, il serait nécessaire d'utiliser une équivalence birationnelle.

Encodage sur des courbes de Montgomery (2013) et des courbes quartiques de Jacobi (Jacobi Quartic Curves) (2015)

• **Courbes de Montgomery :** Yu et al., in 2013 [YWLT13] ont proposé quatre encodages déterministes sur les familles de courbes de Montgomery $E_{a,b} : by^2 = x^3 + ax^2 + x$ définis sur $\mathbb{F}_q$ où q est une puissance d'un nombre premier pair. L'un des encodages est basé sur la recherche de racines cubiques alors que les trois autres sont basés sur la recherche de racines carrées.

• **Quartique de Jacobi :** Yu et al. [YWL+15], in 2015, ont proposé deux encodages déterministes d'un corps fini $\mathbb{F}_q$ sur les quartiques de Jacobi $E_b : y^2 = x^4 + 2bx^2 + 1$. Quand $q \equiv 3(\mod 4)$, leur premier encodage déterministe basé sur l'égalité de Skalba permet d'économiser deux carrés dans un corps fini par rapport à l'équivalence birationnelle composée avec la version simplifiée de Fouque et Tibouchi de la fonction d'encodage

d'Ulas. Quand $q \equiv 2(\bmod 3)$, leur second encodage déterministe basé sur le calcul de la racine cubique coûte une inversion de corps de moins que l'équivalence birationnelle composée avec la fonction d'Icart au prix de quatre multiplications et d'une élévation au carré dans un corps fini.

Encodage sur le modèle d'Intersection de Jacobi (2018)

À Inscrypt-2017, He et al. [HYW17] ont proposé deux encodages sur des familles d'Intersections de Jacobi tordues $E_{a,b} : au^2 + v^2 = bv^2 + w^2 = 1$. Leur premier encodage est basé sur l'encodage SWU et le second sur le cube root encoding. De plus, ils ont estimé la densité des images des deux encodages par le théorème de Tchebotarev [HYW17]. Nous rappelons leur encodage utilisant des racines cubiques (cube root encoding).

Supposons que $q = p^n \equiv 2 \bmod 3$ est une puissance d'un nombre premier impair. Leur cube root encoding $f_2 : \mathbb{F}_q \to E_{a,b} : t \mapsto (u, v, w)$ est défini comme suit.

---

1 **Entrées :** a, b et $t \in: \mathbb{F}_q$ avec $ab(a-b) \neq 0$.
2 **Sortie :** Un point $(u, v, w) \in E_{a,b}(\mathbb{F}_q)$

1. Si $t = 0$ alors retourner $(u, v, w) = (0, 1, 1)$;
2. Sinon poser $\alpha = \frac{a+b+t^2}{3}$, $\beta = \frac{ab-3\alpha^2}{2}$, $\gamma = \alpha t + (t\beta^2 - (\alpha t)^3)^{1/3}$ et retourner $(u, v, w) = \frac{2t}{\gamma^2 - abt^2}(t\gamma + \beta, a(\gamma - bt), b(\gamma - at))$
Notez que puisque $q \equiv 2 \bmod 3$, on peut efficacement calculer la racine cubique par $x^{1/3} = x^{\frac{2q-1}{3}}$.

4.2.2 Encodages sur les courbes hyperelliptiques

Les courbes hyperelliptiques sont des généralisations des courbes elliptiques en genre supérieur (une courbe elliptique est de genre un).

• **Encodage d'Ulas** (*2007*) : Ulas a été le premier à proposer un encodage sur certaines courbes hyperelliptiques [Ula07]. Il a simplifié les résultats (notamment les fonctions rationnelles) de Shalue et Woestjine (voir la section 4.2.1) et les a généralisés pour obtenir des encodages sur les deux familles de courbes hyperelliptiques $y^2 = x^n + ax + b$ et $y^2 = x^n + ax^2 + bx,\ n \geqslant 3, q \equiv 2 \bmod 3$, où $n \geqslant 5$ est un entier positif fixé.

• **KLR-Genus-Two : Encodage de Kammerer-Lercier-Renault** (*2010*) : Dans [KLR10], Kammerer *et al.* ont proposé des encodages pour divers modèles de courbes (hyper)elliptiques, dont, par exemple, le modèle Hessienne $x^3 + y^3 + 1 = 3d$ (voir la section précédente sur les encodages sur les courbes elliptiques), les courbes de genre deux $y^2 = (x^3 + 3ax + 2)^2 + 8b^3$ (type A) et $y^2 = \lambda((x^3 + 3\mu x + 2a)^2 + 4b)$ (type B), sur $\mathbb{F}_q$ (avec $q \equiv 2 \bmod 3$).
Nous renvoyons le lecteur à [KLR10] pour une description plus détaillée de ces encodages, car les formules sont assez compliquées.

• **Encodages unifiés de Seck *et al.***(*2018*) : En suivant la technique d'Elligator-2 [SD18], Seck *et al.* ont proposé dans [SBDK17] trois encodages déterministes *presque*-injectifs sur trois familles de courbes hyperelliptiques données par les équations $\mathbb{H}^i : y^2 = F_i(x)\ \ (i = 1, 2, 3)$, où $F_1(x) = x^5 + ax^4 + cx^2 + dx,\ F_2(x) = x^5 + bx^3 + dx + e,\ F_3(x) =$

$x^5 + ax^4 + e$ sur $\mathbb{F}_q$. Comme beaucoup d'encodages existants (Elligator-2, Encodage sur le modèle de Huff classique, etc.), on peut caractériser l'espace image de leurs encodages $\psi_i : \mathcal{R}_i \to \mathbb{H}^i$ ($\mathcal{R}$ est un certain sous-ensemble de $\mathbb{F}_q$) et leurs encodages peuvent être aussi étendus à $\mathbb{F}_q$ tout entier. Ces travaux ont été récemment unifiés et généralisés par Seck et Diarra [SD18]. En fait, les auteurs ont trouvé des formules unifiées qui fonctionnent avec toute courbe (hyper)elliptique non binaire de la forme $y^2 = h_g(x) = x^{2g+1} + a_{2g-1}x^{2g-1} + a_{2g-}x^{2g-3} + \ldots + a_1x + a_0$, où le genre g de la courbe vérifie $g \leqslant 9$. Nous rappelons ici leur résultat principal.

Considérons la courbe hyperelliptique $\mathbb{H}^g$ de genre g, donnée par

$$y^2 = F_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0 \tag{4.1}$$

sur $\mathbb{F}_q$ ($q = p^n$, $p \neq 2$ et $p \nmid 2g^2 + g$). Nous nous référons à [SD18] pour les valeurs explicites des coefficients a_i. Soit $\alpha_g = 2^{2g-1} - 1$, $\beta_g = 4g^2 + 2g$ et définissons m_g et n_g comme suit :

$$\begin{cases} m_g = \dfrac{\alpha_g\beta_g}{2},\ n_g = (2g^2+g)^2 & \text{si } g \text{ est impair,} \\ m_g = \dfrac{\alpha_g\beta_g}{4},\ n_g = \dfrac{(2g^2+g)^2}{2} & \text{si } g \text{ est pair.} \end{cases}$$

On pose $\mathcal{R}_g = \{r \in \mathbb{F}_q^* :\ F_g(w(ur^2(-m_gs - n_g) - 1)) \neq 0\}$. L'encodage sur la courbe Hyperelliptique $\mathbb{H}^g$ est donné par l'algorithme suivant.

Algorithme 15 : Genus-g-Encoding

1 **Entrée :** La courbe hyperelliptique $\mathbb{H}_g$, et $r \in \mathcal{R}_g$;
2 **Sortie :** Un point (x, y) sur $\mathbb{H}_g$;

1. $v := v(g) = w[ur^2(-m_gs - n_g) - 1]$;

2. $\varepsilon := \chi_q(v^{(2g+1)} + a_{(2g-1)}v^{(2g-1)} + a_{(2g-3)}v^{(2g-3)} + \ldots + a_1v + a_0)$;

3. $x := \dfrac{1+\varepsilon}{2}v + \dfrac{1-\varepsilon}{2}\left(\dfrac{w(-v+w)}{v+w}\right)$;

4. $y := -\varepsilon\sqrt{x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0}$;

5. Retourner $(x, y) \in \mathbb{H}_g$.

Comme toutes les fonctions de type Elligator existantes, cet encodage généralisé est presque injectif et inversible. Nous l'utiliserons dans la dernière section pour définir une fonction de hachage (généralisée) indifférentiable sur toute courbe hyperelliptique $\mathbb{H}^g$ donnée par l'équation (4.1).

4.2.3 Analyses sommaires et comparatives

Il ressort des sections ci-dessus que l'on peut classer presque tous les encodages existants en 4 types (suivant la construction utilisée pour définir l'encodage) : Les encodages de type Icart, les encodages de type Elligator 2, les encodages de type SWU et les encodages Injectifs (injective-like encoding). On obtient alors le tableau suivant.

TABLE 4.1 – Classification des encodages sur les courbes elliptiques et hyperelliptiques

Encodage	Courbe	$\mathbb{F}_q$	SWU	Icart	Injective	Elligator-2
Brier *et al.* [BCI+10]	$y^2 = x^3 + a + b$	$q \equiv 2 \bmod 3$	✓			
Fouque et Tibouchi [FT12]	$y^2 = x^3 + b$	$q \equiv 1 \bmod 3$	✓			
Farashahi [Far11]	$x^3 + y^3 + 1 = 3dxy$	$q \equiv 2 \bmod 3$		✓		
Kammerer *et al.* [KLR10]	$x^3 + y^3 + 1 = 3dxy$	$q \equiv 2 \bmod 3$		✓		
Bernstein *et al.* [BHKL13]	$x^2 + y^2 = 1 + d^2y^2$	$q \equiv 3 \bmod 4$			✓	
Diarra *et al.* [DDK17]	$x^2 + y^2 = 1 + dx^2y^2$	$q \neq 2^n$				✓
He *et al.* [HYW15]	$ax(y^2 - c) = by(x^2 - d)$	$q \equiv 3 \bmod 4$	✓			
	$ax(y^2 - c) = by(x^2 - d)$	$q \equiv 2 \bmod 3$		✓		
Diarra *et al.* [DDK17]	$x(ay^2 - 1) = y(bx^2 - 1)$	$q \equiv 3 \bmod 4$			✓	
	$\alpha x(y^2 - 1) = \beta y(x^2 - 1)$	$q \neq 2^n$				✓
Yu *et al.* [YWLT13]	$by^2 = x^3 + ax^2 + x$	$q \equiv 2 \bmod 3$		✓		
		$q \equiv 3 \bmod 4$	✓			
Yu *et al.* [YWL+15]	$y^2 = x^4 + 2bx^2 + 1$	$q \equiv 2 \bmod 3$		✓		
		$q \equiv 3 \bmod 4$	✓			
He *et al.* [HYW17]	$au^2 + v^2 = bv^2 + w^2 = 1$	$q \equiv 2 \bmod 3$		✓		
		$q \equiv 3 \bmod 4$	✓			
Ulas [Ula07]	$y^2 = x^n + ax + b$ $y^2 = x^n + ax^2 + bx$		✓			
Kammerer *et al.* [KLR10]	$y^2 = (x^3 + 3ax + 2)^2 + 8b^3$	$q \equiv 2 \bmod 3$		✓		
	$y^2 = \lambda((x^3 + 3\mu x + 2a)^2 + 4b)$	$q \equiv 2 \bmod 3$		✓		
Seck *et al.* [SBDK17]	$y^2 = x^5 + bx^3 + dx + e$	$q \equiv 7 \bmod 8$				✓
	$y^2 = x^5 + ax^4 + e$	$q = 5^n$				✓
	$y^2 = x^5 + ax^4 + cx^2 + dx$	$q \equiv 3 \bmod 4$				✓
Seck et Diarra [SD18]	$y^2 = F_g(x),\ g \leqslant 5$	$q \equiv 7 \bmod 8$				✓

4.3 Etat de l'art sur le hachage sur les courbes (hyper)elliptiques

Par une fonction de hachage dans une courbe (hyper)elliptique, nous entendons une fonction $H : \{0,1\}^* \to \mathbb{J}(\mathbb{F}_q)$, qui envoie une chaîne de bits à un élément de la Jacobienne $\mathbb{J}(\mathbb{F}_q)$ de la courbe. Mais pourquoi définir de telles fonctions ? Simplement parce que de nombreux schémas basés sur des courbes elliptiques nécessitent de hacher dans le groupe de points d'une courbe elliptique. Par exemple, Boneh et Franklin [BF01] en 2001 ont été parmi les premiers à exiger un hachage dans les courbes elliptiques ; en fait, la clef publique de leur schéma de chiffrement basé sur l'identité est un point sur la courbe $y^2 = x^3 + b$ (qui est connu pour être supersingulière). Il existe d'autres exemples de schémas utilisant des fonctions de hachage sur des courbes elliptiques, tels que la signature BLS [BLS01], les protocoles d'authentification basés sur des mots de passe (SPEKE[Jab96], PAK[BMP00]), etc.

Dans cette section, nous allons faire une synthèse de toutes les méthodes existantes pour construire des fonctions de hachage sur les courbes elliptiques, en allant de la plus simple (qui bien sûr ne répond pas à toutes les exigences "cryptographiques" des fonctions de hachage) à quelques constructions plus complexes. Et nous verrons comment ces fonctions de hachage sont étroitement liées aux fonctions d'encodage que nous avons vues dans les sections précédentes.

4.3.1 Méthode Try-and-increment (*2001*)

Cette méthode de hachage nécessite à la fois une approche d'encodage trivial (4.2.1) et l'utilisation d'une fonction de hachage cryptographique "classique". Elle a été proposée par Boneh *et al.* dans [BLS01]. En fait, leur schéma de signature (GDH) nécessitait une fonction de hachage $H : \{0,1\}^* \rightarrow G$ qui envoie un message $M \in \{0,1\}^*$ à un point d'une courbe elliptique définie par $y^2 = f(x)$. Ils ont d'abord supposé l'existence de $h : \{0,1\}^* \rightarrow \mathbb{F}_q \times \{0,1\}$, une fonction de hachage cryptographique classique. Et pour hacher un message $m \in \{0,1\}^*$, ils ont procédé comme suit :

1. on met un compteur $c = 0$;
2. on calcule $h(c||m) = (x, b)$;
3. on calcule $f(x)$
4. si $f(x)$ est un carré, alors on retourne $H(m) = (x, (f(x))^{1/2})$ (le bit b permet de choisir une des deux racines carrées de $f(x)$) ; sinon, on incrémente c et on passe à l'étape 2.

Cette méthode de hachage a l'inconvénient de ne pas fonctionner en temps constant : en effet, le temps de fonctionnement dépend du message m à hacher. Cela pourrait conduire à des attaques de canaux cachés.

4.3.2 La Construction $H = f \circ h$ [Brier et al. (*2010*)]

Brier *et al.* [BCI+10] ont généralisé et formalisé la notion d'*encodage admissible* par Boneh et Franklin [BF01] et ont montré que la construction $H = f \circ h$ est indifférentiable d'un oracle aléatoire, où f est un encodage admissible et h est une fonction de hachage (classique) modélisée comme un oracle aléatoire. Mais la plupart des encodages déterministes connus ne sont pas admissibles (être admissible inclut être une fonction $r : 1$) puisque leur ensemble image n'est qu'une fraction de la courbe (environ 5/8 pour l'encodage Icart par exemple). Ainsi, cette construction ne peut pas remplacer un oracle aléatoire dans le groupe de points de la courbe, sauf pour certains cas particuliers (comme l'encodage sur une courbe supersingulière utilisé par Boneh et Franklin dans leur schéma IBE [BF01], qui est une bijection de $\mathbb{F}_q$ au groupe de points rationnels de la courbe,).

4.3.3 La Construction $H = f \circ h_1 + h_2\mathbb{G}$ [Brier *et al.* (*2010*)]

La construction précédente de la fonction de hachage indifférentiable (utilisant des encodages admissibles) ne fonctionne pas pour la plupart des encodages déterministes existants, puisqu'ils ne peuvent pas être admissibles. Les auteurs ont donc proposé une autre méthode de hachage, qui utilise la notion de weak-encodage (qui est plus légère que la notion d'admissibilité, et couvre donc plus d'encodages). Considérons une courbe elliptique E sur $\mathbb{F}_q$ telle que $E(\mathbb{F}_q)$ est cyclique d'ordre n, et un weak-encodage $f : \mathbb{F}_q \rightarrow E(\mathbb{F}_q)$. Soit $h_1 : \{0,1\}^* \rightarrow \mathbb{F}_q$ et $h_2 : \{0,1\}^* \rightarrow \mathbb{Z}/n\mathbb{Z}$ deux fonctions de hachage vues comme des oracles aléatoires. Alors la construction $H(m) = f(h_1(m)) + h_2(m)G$ est indifférentiable d'un oracle aléatoire dans la courbe.
Cette construction est plus générale que la précédente mais elle est moins efficace car elle nécessite une multiplication par un scalaire sur la courbe.

4.3.4 La Construction $H = f \circ h_1 + f \circ h_2$ [Brier *et al.*(*2010*)]

Brier *et al.* [BCI+10], en utilisant la notion d'*encodage admissible*, ont montré que la construction $H(m) = f(h_1(m)) + f(h_2(m))$ est indifférentiable à un oracle aléatoire, où f est l'encodage de l'Icart 4.2.1 et h_1, h_2 sont deux fonctions de hachage (classiques) modélisées comme des oracles aléatoires.
La preuve implique des encodages bien distribués et des sommes de caractères.

4.3.5 Généralisation de $H = f \circ h_1 + f \circ h_2$ [Farashahi *et al.* (*2013*)]

Farashahi *et al.* [FFS+13] ont généralisé la construction de Brier *et al.* à tous les encodages déterministes connus (et bien distribués) sur les courbes elliptiques et hyperelliptiques. En fait, ils ont proposé la construction $H(m) = f(h_1(m)) + \ldots + f(h_s(m))$, qui préserve l'indifférentiabilité si les h_i sont modélisés comme des oracles aléatoires, f est un encodage bien distribué sur la courbe et s est un entier strictement supérieur au genre de la courbe. Ceci permet de hacher dans n'importe quelle courbe Hyperelliptique où un encodage bien distribué est connu. De nombreux travaux ont utilisé plus tard cette construction pour construire des fonctions de hachage sur des courbes elliptiques, comme dans [DDK17, SBDK17, HYW17, YWL+15].

Après avoir énuméré toutes les méthodes de hachage existantes, nous sommes maintenant en mesure de donner la classification suivante des fonctions de hachage sur les courbes (hyper)elliptiques, comme nous l'avons fait pour classer les encodages sur les courbes (hyper)elliptiques.

TABLE 4.2 – Fonctions de hachage indifférentiables existantes sur les courbes (hyper)elliptiques

Hachage sur	**Méthode de Hachage**	**Encodage utilisé**
$E_b: \ y^2 = x^3 + b$	$H = f \circ h$	Boneh and Franklin [BF01]
$E_b: \ y^2 = x^3 + b$	$H = f \circ h_1 + \ldots + f \circ h_s$	Fouque and Tibouchi [FT12]
$E_d: \ x^2 + y^2 = 1 + dx^2y^2$	$H = f \circ h_1 + \ldots + f \circ h_s$	Diarra et al. [DDK17]
$E_{a,b} : y^2 = x^3 + ax + b$	$H = f \circ h_1 + h_2\mathbb{G}$ $H = f \circ h_1 + h_2\mathbb{G}$	Brier et al. [BCI+10]
$E_{a,b} = y^2 = x^3 + ax + b$	$H = f \circ h_1 + \ldots + f \circ h_s$	Farashahi et al. [FFS+13]
$H = x^3 + (y + c)(3x + 2a + \frac{2b}{y}) = 0$	$H = f \circ h_1 + \ldots + f \circ h_s$	Farashahi et al. [FFS+13]
$E_1: \ ax(y^2 - c) = by(x^2 - d)$	$H = f \circ h_1 + h_2\mathbb{G}$	He et al. [HYW15]
$E_b = y^2 = x^4 + 2bx^2 + 1$	$H = f \circ h_1 + \ldots + f \circ h_s$	Yu et al. [YWL+15]
$E_{a,b} : by^2 = x^3 + ax^2 + x$	$H = f \circ h_1 + h_2\mathbb{G}$ $H = f \circ h_1 + \ldots + f \circ h_s$	Yu et al. [YWLT13]
$E_{a,b} : au^2 + v^2 = bv^2 + w^2 = 1$	$H = f \circ h_1 + \ldots + f \circ h_s$	He et al. [HYW17]
$\mathbb{H}^1 : y^2 = x^5 + ax^4 + cx^2 + dx$ $\mathbb{H}^2 : y^2 = x^5 + a_3x^3 + a_1x + a_0$	$H = f \circ h_1 + \ldots + f \circ h_s$	Seck et al. [SBDK17]
$\mathbb{H}_g : y^2 = x^{2g+1} + a_{2g-1}x^{2g-1} + \ldots + a_1x + a_0$	$H = f \circ h_1 + \ldots + f \circ h_s$	Seck et al. [SD18]

4.4 Formules unifiées pour le hachage sur des courbes (hyper)elliptiques

Cette partie utilise l'encodage généralisé de Seck et Diarra dans AFRICACRYPT 2018. Notre but principal, dans cette section, est de montrer qu'on peut concevoir une fonction de hachage indifférentiable sur la Jacobienne de la courbe hyperelliptique $\mathbb{H}_g$: $y^2 = F_g(x) = x^{2g+1} + a_{2g-1}x^{2g-1} + a_{2g-}x^{2g-3} + \ldots + a_1x + a_0$, $g \leqslant 9$ en utilisant l'encodage unifié ψ_g de Seck et Diarra [SD18] et le framework de Farashahi *et al.* [FFS+13].

Rappelons quelques résultats de Seck et Diarra. On définit

— $\alpha_g = 2^{2g-1} - 1$, $\beta_g = 4g^2 + 2g$;

— $m_g = \dfrac{\alpha_g\beta_g}{2}$, $n_g = (2g^2+g)^2$ si g est impair; et $m_g = \dfrac{\alpha_g\beta_g}{4}$, $n_g = \dfrac{(2g^2+g)^2}{2}$ si g est pair;

— $\mathcal{R}_g = \{r \in \mathbb{F}_q^* : \ F_g(w(ur^2(-m_gs - n_g) - 1)) \neq 0\}$ et $S_g = \mathbb{F}_q \backslash \mathcal{R}_g$ où $w \in \mathbb{F}_q^\star$ est un paramètre arbitraire, $u \in \mathbb{F}_q$ est un paramètre non nul et non carré et $s \in \mathbb{F}_q$ vérifie l'équation $\alpha_g s^2 + \beta_g s - \gamma_g = 0$. L'ensemble S_g contient au plus $2(2g+1)+1$ éléments.

L'encodage unifié est défini comme suit : $\psi_g : \mathcal{R}_g \to \mathbb{H}_g : r \mapsto \psi_g(r) = (x, y)$ où

$x = \frac{1+\varepsilon}{2}v + \frac{1-\varepsilon}{2}\left(\frac{w(-v+w)}{v+w}\right)$ and $y = -\varepsilon\sqrt{F_g(x)}$ avec $v = w[ur^2(-m_gs-n_g)-1]$ et $\varepsilon = \chi_q(v^{(2g+1)} + a_{(2g-1)}v^{(2g-1)} + a_{(2g-3)}v^{(2g-3)} + \ldots + a_1v + a_0)$.

Extension de ψ_g à $\mathbb{F}_q$

On se propose d'étendre l'encodage ψ_g à $\mathbb{F}_q$ comme suit : On choisit $w \in \mathbb{F}_q^*$, de telle sorte que $F_g(w)$ soit un carré non nul.

— $\psi_g(0) = (-w, \sqrt{F_g(-w)})$.

— On a $v(r) = w[ur^2(-m_gs - n_g) - 1]$. On pose $Z_g = \left\{r \in \mathbb{F}_q^* : F_g(v(r)) = 0\right\}$. Si Z_g n'est pas vide, on pose $\psi_g(\pm r) = (v(r), 0)$ pour tout $r \in Z_g$ pour conserver la presque injectivité (on pouvait aussi poser $\psi_g(\pm r) = (-w, -\sqrt{F_g(-w)})$ $\forall r \in Z_g$ avec le choix ci-dessus de w).

Théorème 4.4.2 (Seck et Diarra [SD18])

1. Soit (x, y) un point de la courbe hyperelliptique $\mathbb{H}_g$. Alors $(x, y) \in \mathrm{Im}(\psi_g)$ si et seulement si $uw(x + w)(-n_g - m_gs)$ est un carré non nul dans $\mathbb{F}_q$.
2. Soit $(x, y) \in \mathrm{Im}(\psi_g)$ et on définit $\bar{r}$ comme suit :
   $\bar{r} = \sqrt{\frac{x+w}{uw(-n_g-m_gs)}}$ si $y \notin \sqrt{\mathbb{F}_q^2}$ et $\bar{r} = \sqrt{\frac{2w}{u(x+w)(-n_g-m_gs)}}$ si $y \in \sqrt{\mathbb{F}_q^2}$.

   Alors $\bar{r} \in \mathcal{R}_g$ et $\psi_g(\bar{r}) = (x, y)$.

D'après le théorème précédent, nous savons que

$\chi_q(y) = -1 \iff y \notin \sqrt{\mathbb{F}_q^2} \iff f_1(x, r) = uw(-n_g - m_gs)r^2 - (x + w) = 0$

$\chi_q(y) = 1 \Leftrightarrow y \in \sqrt{\mathbb{F}_q^2} \Leftrightarrow f_2(x, r) = u(x + w)(-n_g - m_gs)r^2 - 2w = 0$

On définit les recouvrements $h_j : C_j \to \mathbb{H}_g, j = 1, 2$, par la courbe projective lisse dont les corps de fonction sont les extensions (désignées par $\mathbb{F}_q(x, y, r)$) de $\mathbb{F}_q(x, y)$ définies par $uw(-n_g - m_gs)r^2 - (x + w) = 0$ et $u(x + w)(-n_g - m_gs)r^2 - 2w = 0$ respectivement. En

d'autres termes, un point rationnel dans $C_j(\mathbb{F}_q)$ est un 3-uplet (x, y, r) tel que $(x, y) \in \mathbb{H}_g$ et $f_j(x, r) = 0$. En particulier, pour tout $r \in \mathcal{R}_g$, il y a deux points rationnels de $\mathbb{H}_g$ dont la troisième coordonnée est r.En utilisant ce résultat, on peut définir des morphismes $g_j : C_j \to \mathbb{P}^1$, tels que tout point dans $\mathbb{A}^1(\mathbb{F}_q)\backslash S_g$ possède exactement deux pré-images dans $C_j(\mathbb{F}_q)$ pour l'un de $j = 1, 2$, et aucune dans l'autre. Ces deux pré-images sont conjuguées via $y \mapsto -y$, de sorte qu'exactement une d'entre elles satisfait $\chi_q(y) = \left(\frac{y}{q}\right) = (-1)^j$ puisque $q \equiv 7 \bmod 8$ et donc $q \equiv 3 \bmod 4$. Si on désigne par $Q \in C_j(\mathbb{F}_q)$ cette pré-image, alors $\psi_g(r) = h_j(Q)$.

Théorème 4.4.3
Pour tout caractère non trivial χ de $\mathrm{Jac}(\mathbb{F}_q)$ de $\mathbb{H}_g(\mathbb{F}_q)$, la somme des caractères $S_{\psi_g}(\chi)$ satisfait :

$$\left|S_{\psi_g}(\chi)\right| \leqslant (16g)\sqrt{q} + (44g + 31)$$

où g est le genre de $\mathbb{H}_g$.

Preuve : Nous savons que

$$\begin{aligned}\left|S_{\psi_g}(\chi)\right| = \left|\sum_{r\in\mathbb{F}_q} \chi(\psi_g(r))\right| = & \left|\sum_{r\in\mathcal{R}_g} \chi(\psi_g(r)) + \sum_{r\in S_g} \chi(\psi_g(r))\right| \\ \leqslant & \left|\sum_{r\in\mathcal{R}_g} \chi(\psi_g(r))\right| + \#S_g\end{aligned}$$

où $\#A$ désignent le cardinal de A. On a

$$\sum_{r\in\mathcal{R}_g} \chi(\psi_g(r)) = \sum_{\substack{Q\in C_1(\mathbb{F}_q)\backslash S_{g,1}\\ \chi_q(y)=-1}} \chi(h_1(Q)) + \sum_{\substack{Q\in C_2(\mathbb{F}_q)\backslash S_{g,2}\\ \chi_q(y)=1}} \chi(h_2(Q))$$

où $S_{g,j} = g_j^{-1}(S_g \cup \{\infty\})$. Donc

$$\left|\sum_{r\in\mathcal{R}_g} \chi(\psi_g(r))\right| \leqslant \left|\sum_{\substack{Q\in C_1(\mathbb{F}_q)\\ \chi_q(y)=-1}} \chi(h_1(Q))\right| + \left|\sum_{\substack{Q\in C_2(\mathbb{F}_q)\\ \chi_q(y)=1}} \chi(h_2(Q))\right| + \#S_{g,1} + \#S_{g,2}$$

Pour estimer chaque somme dans la partie droite de l'inégalité précédente, nous utiliserons l'égalité suivante

$$\sum_{Q\in C_j(\mathbb{F}_q)} \chi(h_j(Q)).\left(\frac{(1 + (-1)^j\chi_q(y)}{2}\right) = \sum_{\substack{Q\in C_j(\mathbb{F}_q)\\ \chi_q(y)=(-1)^j}} \chi(h_j(Q)) + \frac{1}{2}\sum_{\substack{Q\in C_j(\mathbb{F}_q)\\ \chi_q(y)=0}} \chi(h_j(Q))$$

Nous en déduisons que

$$\sum_{\substack{Q\in C_j(\mathbb{F}_q)\\ \chi_q(y)=(-1)^j}} \chi(h_j(Q)) = \sum_{Q\in C_j(\mathbb{F}_q)} \chi(h_j(Q)).\left(\frac{(1 + (-1)^j\chi_q(y)}{2}\right) - \frac{1}{2}\sum_{\substack{Q\in C_j(\mathbb{F}_q)\\ \chi_q(y)=0}} \chi(h_j(Q))$$

— D'une part, la somme $\sum_{\substack{Q\in C_j(\mathbb{F}_q)\\ \chi_q(y)=0}} \chi(h_j(Q))$ comporte au plus $2\times(2g+1)=4g+2$ termes où g est le genre de la courbe hyperelliptique $\mathbb{H}_g$.

— D'autre part, $h_j, j=1,2$ sont totalement ramifiés sur les points de $\mathbb{H}_g$ tels que $x=-w$, donc ils ne peuvent se factoriser à travers d'un recouvrement non ramifié de $\mathbb{H}_g$. En appliquant l'inégalité 2.1 dans le Théorème (2.2.8, Chapitre 2), nous avons

$$\left|\sum_{Q\in C_j(\mathbb{F}_q)} \chi(h_j(Q)).\left(\frac{(1+(-1)^j\chi_q(y)}{2}\right)\right| \leqslant (2g_{C_j}-2-\deg y)\sqrt{q}$$

où g_{C_j} est le genre de C_j et $\deg y$ est le degré de y comme fonction rationnelle sur C_j. Puisque $[\mathbb{F}_q(x,y,r):\mathbb{F}_q(x,y)]=2$ et $[\mathbb{F}_q(x,y):\mathbb{F}_q(y)]=2g+1$, alors $\deg y=[\mathbb{F}_q(x,y,r):\mathbb{F}_q(y)]=2\times(2g+1)=4g+2$. Maintenant, nous allons déterminer le genre g_{C_j} de la courbe C_j

Le recouvrement $h_j: C_j\to\mathbb{H}_g$ est seulement ramifié aux points avec $x=-w$. Donc, par la formule de Riemann-Hurwitz, on obtient

$$2g_{C_j}-2=2(2g-2)+2(2-1)=4g-2\Longleftrightarrow g_{C_j}=2g$$

Finalement $\left|\sum_{Q\in C_j(\mathbb{F}_q)}\chi(h_j(Q)).\left(\frac{(1+(-1)^j\chi_q(y)}{2}\right)\right|\leqslant(8g)\sqrt{q}$.

Et donc $\left|\sum_{\substack{Q\in C_j(\mathbb{F}_q)\\ \chi_q(y)=(-1)^j}}\chi(h_j(Q))\right|\leqslant(8g)\sqrt{q}+(2g+1)$

On en déduit que

$$\begin{aligned}\left|\sum_{r\in\mathcal{R}_g}\chi(\psi_g(r))\right|\leqslant\quad&\left((8g)\sqrt{q}+(2g+1)+\#S_{g,1}+\frac{1}{2}\#S_g\right)+\\&\left((8g)\sqrt{q}+(2g+1)+\#S_{g,2}+\frac{1}{2}\#S_g\right)\\\leqslant\ &(16g)\sqrt{q}+(4g+2)+\#S_{g,1}+\#S_{g,2}+\#S_g\end{aligned}$$

Nous savons que $\#S_g\leqslant 8g+5$ et $S_{g,j}\leqslant 2(\#S_g+1)\leqslant 16g+12$ puisque g_j est une application de degré 2. Ainsi

$$\left|\sum_{r\in\mathcal{R}_g}\chi(\psi_g(r))\right|\leqslant(16g)\sqrt{q}+(4g+2)+32g+24+8g+5=(8g+16)\sqrt{q}+(44g+29)$$

Nous obtenons enfin $\left|S_{\psi_g}(\chi)\right|\leqslant 16g\sqrt{q}+(44g+31)$. En particulier, nous avons

— $|S_{\psi_2}(\chi)|\leqslant 32\sqrt{q}+119$ si $g=2$ (c'est la borne obtenue par Seck *et al.* [SBDK17])

— et si $g=3$ alors $|S_{\psi_3}(\chi)|\leqslant 48\sqrt{q}+163$.

Nous pouvons conclure que l'encodage ψ_g est bien distribué en utilisant le Théorème 2.2.7. Et par conséquent la fonction de hachage $H(m)=\psi_g(h_1(m))+\ldots+\psi_g(h_s(m)), s>2$ est indifférentiable d'un oracle aléatoire quand $h_1,\ldots,h_s$ sont vus comme des oracles aléatoires à $\mathbb{F}_q$ d'après Farashahi *et al.* [FFS+13].

□

4.5 Conclusion

Nous avons donné l'état de l'art de tous les encodages existants et des fonctions de hachage indifférentiables sur les courbes elliptiques et hyperelliptiques. Nous avons également construit une fonction de hachage générique indifférentiable pour certaines familles de courbes hyperelliptiques, en utilisant les formules unifiées de Seck et Diarra [SD18] et le framework de Farashahi *et al* [FFS+13].

Deuxième partie

SimulaMath : un logiciel de calcul

Chapitre 5

Le logiciel SimulaMath [1]

Contents

1. Notez que toutes les figures présentées dans cette thèse sont réalisées avec SimulaMath : un logiciel de calcul scientifique que nous avons conçu.

SimulaMath (https://simulamath.org) est un logiciel de calcul scientifique que nous avons développé avec Python. Il est conçu en mettant l'accent sur la puissance et la facilité d'utilisation, par le biais d'une interface utilisateur graphique (GUI : Graphical User Interface en anglais). Il couvre de nombreux domaines des mathématiques comme l'algèbre linéaire, l'analyse, la théorie des nombres, la statistique descriptive, les distributions de probabilités, les graphiques en 2D et 3D, les bases de Groebner, les réseaux arithmétiques, les courbes elliptiques et les codes linéaires. Il fonctionne normalement sur les plates-formes Windows, Mac OSX et Linux.

5.1 Introduction

SimulaMath est développé avec le langage de programmation python [Ros95]. Le projet SimulaMath a été lancé par moi même fin 2015. La version 1.0 beta 1 a été publiée en 2017 et la 1.0 beta 2 en 2018. Lorsque le projet a été lancé, l'objectif principal était de concevoir un logiciel qui pourrait faciliter la recherche, l'enseignement et l'apprentissage en mathématiques pour différents niveaux (collège, lycée et université). Pour atteindre cet objectif, le logiciel devrait être facile à utiliser, couvrir de nombreux domaines des mathématiques et être compétitif avec les logiciels disponibles dans le domaine (de l'éducation) comme Geogebra [HBA+18].

Python [Ros95] est un langage de programmation puissant, facile à apprendre et à utiliser. Le typage dynamique de Python, sa syntaxe élégante, sa communauté qui s'augmente rapidement, ses différents modules, ainsi que sa nature interprétée, en font de lui un langage idéal pour de nombreux développeurs professionnels et chercheurs scientifiques. Sa simplicité, la lisibilité du code, en font de lui un choix approprié dans l'enseignement pour tous les niveaux. On peut utiliser Python pour diverses applications : le développement web avec Django et Flask, le développement mobile avec Kivy, le développement d'interfaces graphiques avec Tkinter, PyQt, WxPython et Kivy, la science des données avec Sklearn et Pandas, les graphiques avec Matplotlib et Seaborn, les calculs numériques avec Nympy, les calculs symboliques avec Sympy, la statistique et les probabilités avec Scipy et Statsmodel, etc [Ros95, MSP+17, Oli , JOP+ , JOP+ , Hun07]. Toutes ces merveilles font de Python le choix parfait de certains logiciels comme SageMath [S+09] et le présent logiciel SimulaMath que nous décrivons

SageMath [S+09] est un logiciel de mathématiques libre et open-source qui est conçu à partir de nombreux bibliothèques open-sources existantes comme Matplotlib [Hun07], NumPy [Oli], SciPy[JOP+], Sympy [MSP+17], GAP[GAP19], Singular [DGPS19], R[MRS+13] etc. Le but principal du projet SageMath est de fournir une alternative viable, gratuite et open-source à Maple [VWY+13], Mathematica [Inc], Magma [BCP97] et MATLAB [MAT18]. L'objectif du projet SageMath n'est pas seulement de créer un logiciel pour la statistique et les probabilités comme R, la géométrie algébrique comme Singular et ainsi de suite mais pour tous les domaines des mathématiques. On peut utiliser SageMath pour l'algèbre linéaire, la théorie des groupes, les probabilités, la statistique, la géométrie algébrique, l'analyse, la théorie des nombres, la cryptographie etc. L'objectif à long terme de l'équipe de Sage est de faire de ce dernier un logiciel utile, efficace, libre et open

source, facile à compiler et extensible. C'est l'une des raisons pour lesquelles SageMath est largement utilisé par la communauté scientifique. C'est est un excellent choix pour la recherche par contre il est moins efficace pour l'enseignement (du moins pour le collège et le lycée).

GeoGebra [HBA+18] est un logiciel de mathématiques dynamique qui permet de faire de la géométrie interactive, de l'analyse, de la probabilité, des statistiques et de l'algèbre. Il a été initialement développé par Markus Hohenwarter qui a débute le projet en 2001 dans le cadre de son mémoire de maîtrise à l'Université de Salzbourg en Autriche. GeoGebra est disponible en plusieurs langues et est maintenant maintenu par une équipe internationale de programmeurs. Il est surtout utilisé pour l'enseignement et l'apprentissage des mathématiques de l'école primaire à l'école secondaire mais il est peu utilisé dans les universités.

SimulaMath est un logiciel de calcul scientifique et de simulation dont l'objectif principal est de faciliter l'enseignement et l'apprentissage des mathématiques du collège au Master à l'Université, d'une part en Afrique et d'autre part, partout dans le monde. Comme Sage, SimulaMath est conçu à partir des modules scientifiques de Python comme Matplotlib, Numpy, Scipy, Sympy et Pandas. SimulaMath a été conçu, non pas pour réinventer la roue mais pour simplifier davantage ce qui existe. Nous avons déjà cité la puissance du langage de programmation Python comme la lisibilité, la facilité d'utilisation et la puissance du logiciel SageMath. En ce qui concerne SimulaMath, il prend une nouvelle direction qui est de rendre les calculs très simples pour le monde de l'éducation en exploitant la puissance du beau langage de programmation Python et ses modules. L'objectif à long-terme est d'implémenter les programmes de mathématiques africains de niveau moyen et secondaire mais aussi le programme international. Pour l'Université, il s'agira d'implémenter les matières les plus enseignées en mathématiques telles que l'algèbre, l'analyse, les probabilités, la statistique, la théorie des nombres, la cryptographie etc.

Dans la suite, nous faisons une petite comparaison de SimulaMath avec quelques logiciels. Nous montrons également comment utiliser SimulaMath pour faire des calculs et des simulations dans quelques domaines des mathématiques.

5.2 Étude comparative

Dans cette section, nous montrons les particularités du logiciel SimulaMath en faisant une petite comparaison, en particulier, avec SageMath et Geogebra.

SimulaMath vs SageMath : Les deux logiciels sont développés avec Python. SageMath est très puissant en ce qui concerne la recherche dans beaucoup de domaines des mathématiques, mais il n'est pas très approprié pour l'enseignement des mathématiques, en particulier au collège et au lycée. Il est bien documenté et dispose d'une très grande communauté (développeurs et utilisateurs). SimulaMath est un logiciel récent qui se concentre d'abord sur l'enseignement et l'apprentissage des mathématiques et donne un environnement très simple pour la simulation et les calculs. Pour réaliser de bonnes choses avec Sa-

geMath, vous devez être un bon programmeur (en Python), mais SimulaMath ne demande aucun pré-requis en programmation pour la version (1.0). Cependant, pour SimulaMath, il est prévu dans le futur de fournir à la fois une interface cliquable assez complet comme Maple et Geogebra et une interface de programmation simple comme SageMath. Il faut noter que ce n'est pas une vraie comparaison, SageMath n'est pas encore comparable avec SimulaMath en ce qui concerne le contenu et les domaines, mais ceci vous donne juste la philosophie des deux logiciels.

SimulaMath vs. Geogebra : Geogebra est l'un des logiciels dynamiques les plus utilisés dans l'enseignement et l'apprentissage des mathématiques en Afrique de l'Ouest et peut-être dans beaucoup d'autres pays. Avec Geogebra, vous pouvez faire de l'analyse, de l'algèbre, de la probabilité, des statistiques, des graphiques et de la géométrie interactive et pour SimulaMath, on peut aussi faire presque les mêmes choses (parfois plus ou moins efficacement) et d'autres domaines tels que les bases de Groebner, les réseaux arithmétiques, les codes linéaires et les courbes elliptiques. SimulaMath est développé avec Python qui est devenu le langage de programmation pour la science et est le plus souvent enseigné comme premier langage de programmation dans de nombreuses écoles tandis que GeoGebra est développé avec le langage Java. Dans les prochaines versions de SimulaMath, nous avons l'intention d'utiliser Python comme langage de programmation pour le logiciel SimulaMath comme SageMath.

Entrées/sorties SimulaMath : La plupart des logiciels scientifiques n'ont pas d'entrées/sorties triviales. L'un des principaux objectifs de SimulaMath est de simplifier au maximum la syntaxe (les entrées et les sorties). Autrement dit la syntaxe de SimulaMath devrait être très proche de la syntaxe (notation) mathématique.

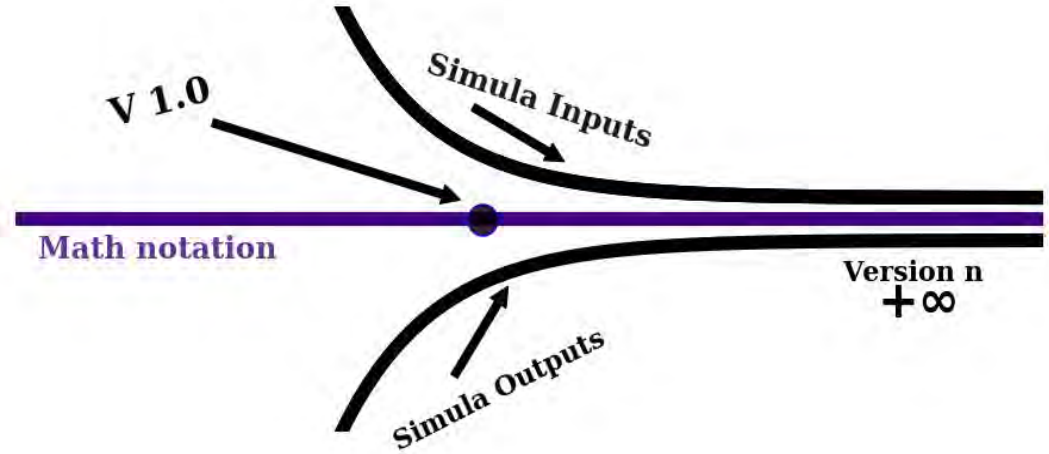
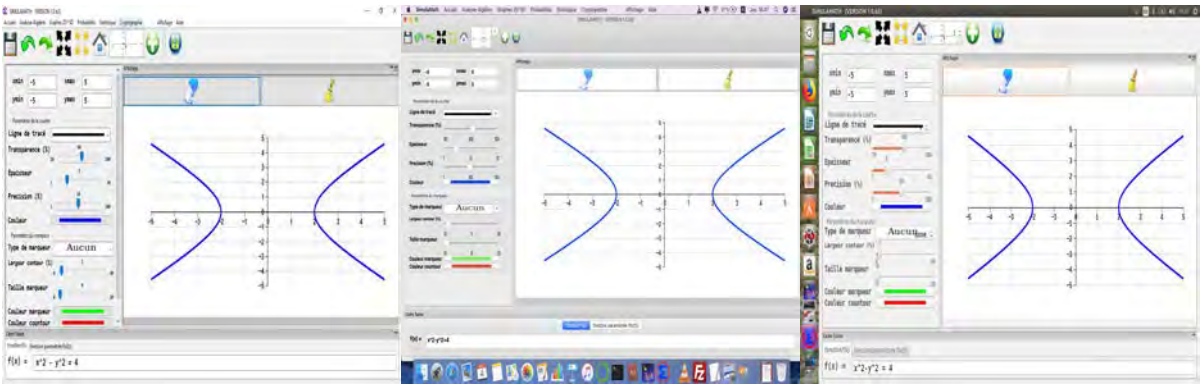


FIGURE 5.1 – Entrées/sorties de SimulaMath et les notations en Maths[4]

4. Cette figure est conçue avec SimulaMath

Liens de téléchargement de SimulaMath : Les versions Windows, Mac OS X et Linux sont disponibles. https://simulamath.org/sim/download.



(a) Windows (b) Mac OS X (c) Linux (Ubuntu)

FIGURE 5.2 – SimulaMath 1.0 dans différentes plateformes

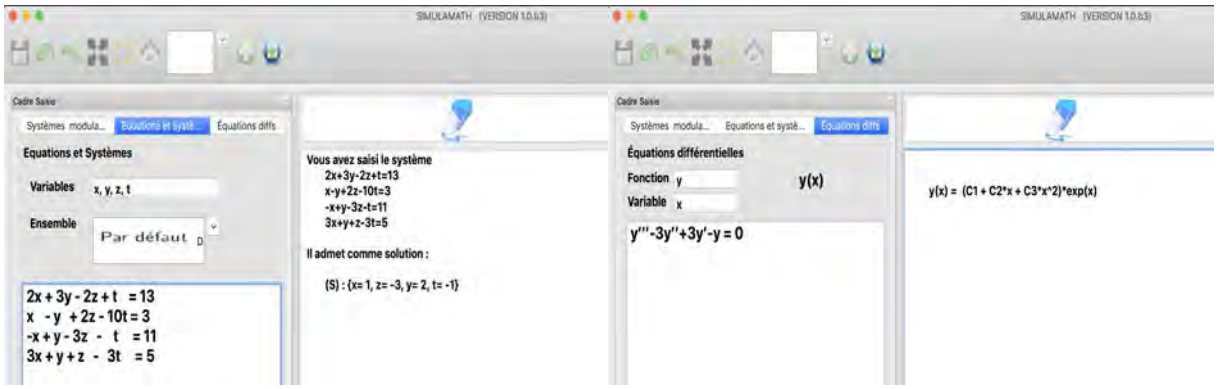
5.3 SimulaMath et l'analyse

Avec SimulaMath, on peut facilement résoudre des équations, des systèmes linéaires et certaines équations différentielles. Soit à résoudre les équations suivantes :

— le système linéaire

$$\begin{cases} 2x + 3y - 2z + t & = 13 \\ x - y + 2z - 10t & = 3 \\ -x + y - y - 3z - t & = 11 \\ 3x + y + z - 3t & = 5 \end{cases}$$

— l'équation différentielle $y''' - 3y'' + 3y' - y' - y = 0$



(a) Système d'équations (b) équation différentielle

FIGURE 5.3 – SimulaMath et résolution d'équations et de systèmes

On peut aussi faire des opérations sur les fonctions comme le calcul d'une intégrale, d'une dérivée, des racines et la factorisation d'une expression.

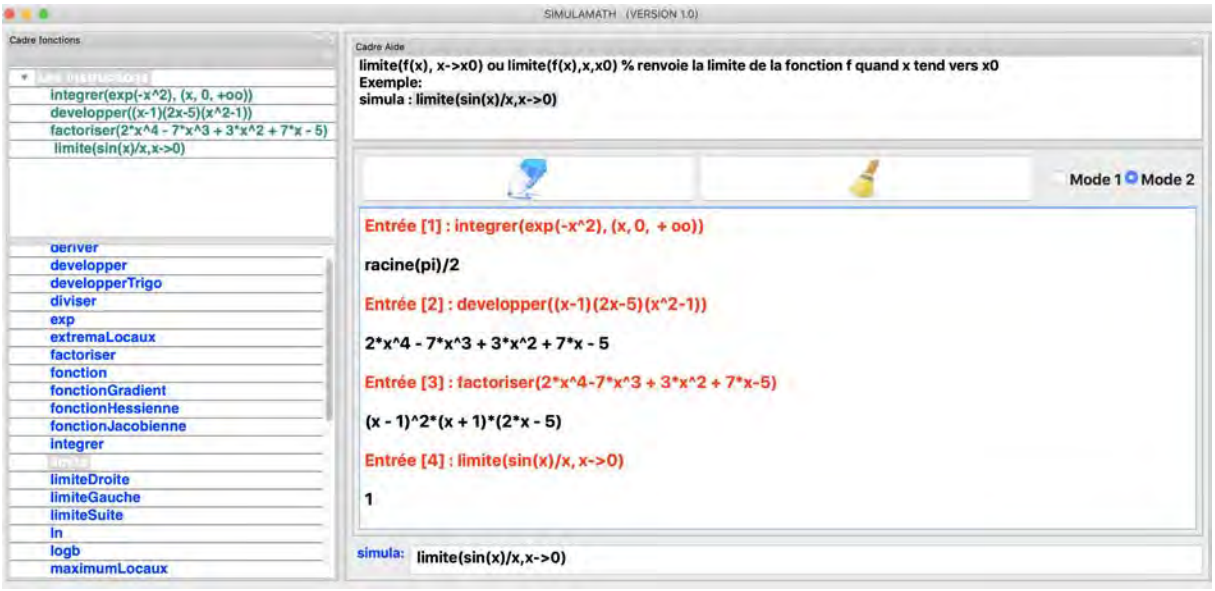


FIGURE 5.4 – SimulaMath et l'analyse

5.4 SimulaMath et les courbes elliptiques

Pour les courbes elliptiques, la forme courte de Weierstrass $y^2 = x^3 + ax + b$ et la forme de Montgomery $By^2 = x^3 + Ax^2 + x$ sur un corps fini $\mathbb{F}_p$ où p est un nombre premier sont implémentés. On peut simuler (tracer) la loi de groupe d'une courbe elliptique sur $\mathbb{R}$ et effectuer certaines opérations sur les points d'une courbe elliptique.

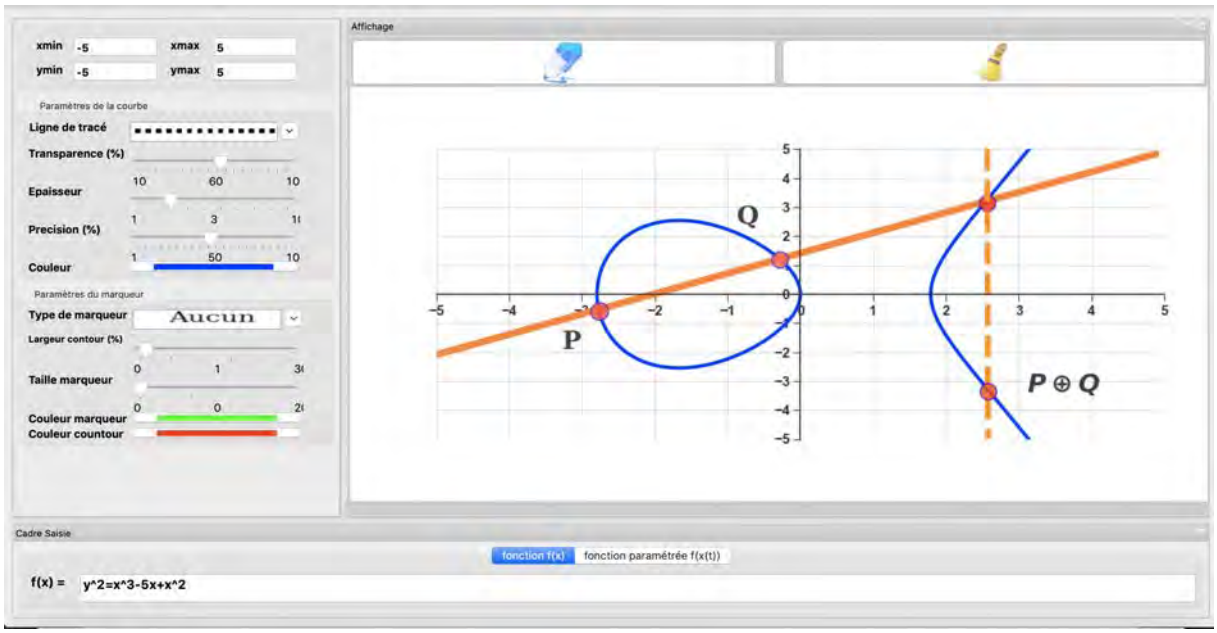


FIGURE 5.5 – SimulaMath et la loi de groupe d'une courbe elliptique

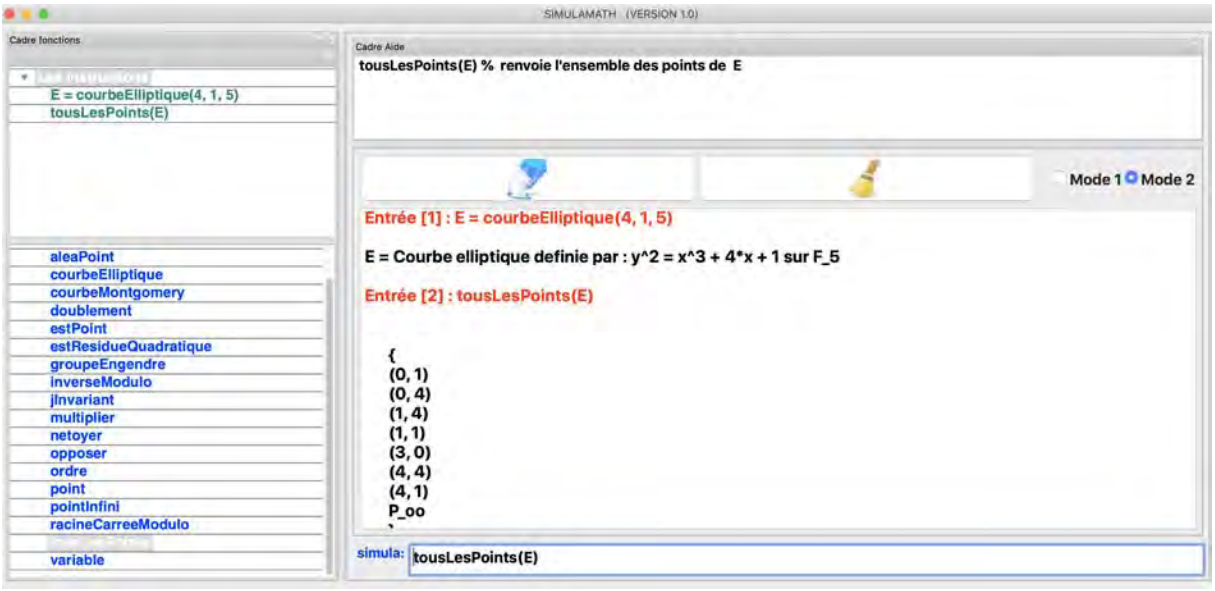


FIGURE 5.6 – SimulaMath et les courbes elliptiques

5.5 Codes linéaires et SimulaMath

Pour la théorie du codes, les codes linéaires incluant les codes de Hamming binaires sont implémentés. On peut définir un code linéaire par sa matrice génératrice ou sa matrice de parité et effectuer certaines opérations.

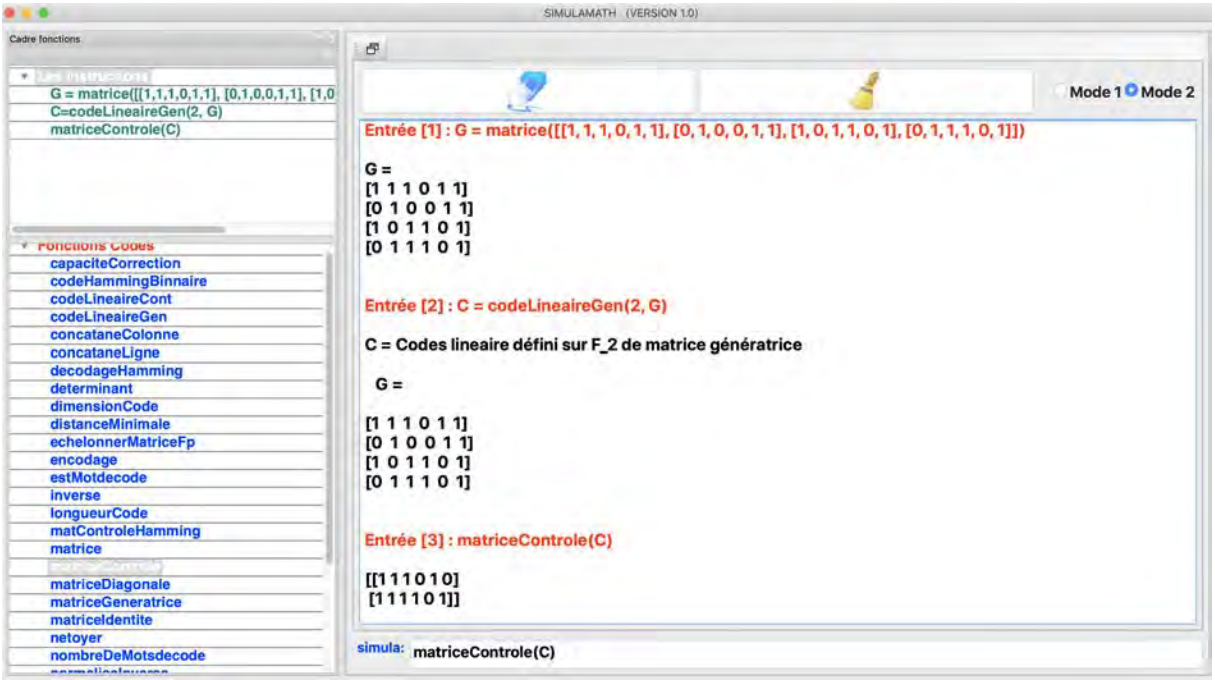


FIGURE 5.7 – SimulaMath et codes linéaires

5.6 SimulaMath et l'algèbre linéaire

Beaucoup d'opérations peuvent être effectuées sur des matrices. Soit la matrice A suivante

$$A = \begin{pmatrix} -1 & 1 & 1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{pmatrix}$$

On peut déterminer le polynôme caractéristique, le déterminant et le rang de A avec SimulaMath comme suit.

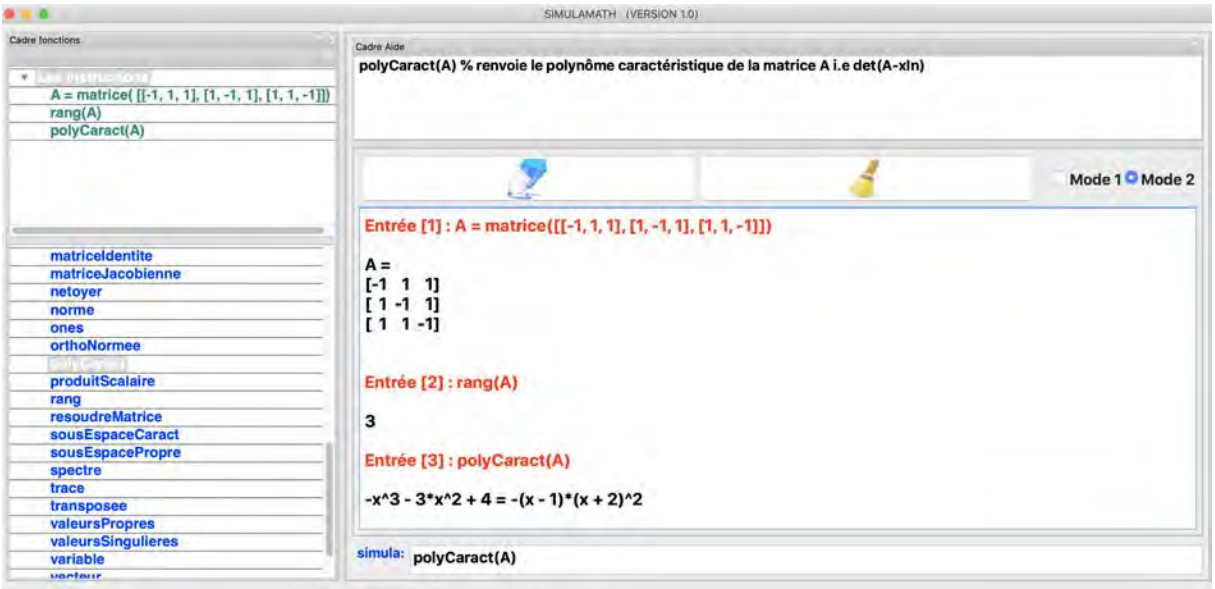


FIGURE 5.8 – SimulaMath et l'algèbre linéaire

5.7 SimulaMath et Graphiques 2D et 3D

Il existe de nombreux types de graphiques disponibles dans SimulaMath, y compris les courbes paramétriques et les diagrammes statistiques tels que les diagrammes à barres, les diagrammes circulaires, les histogrammes, les diagrammes en boîtes et les courbes.

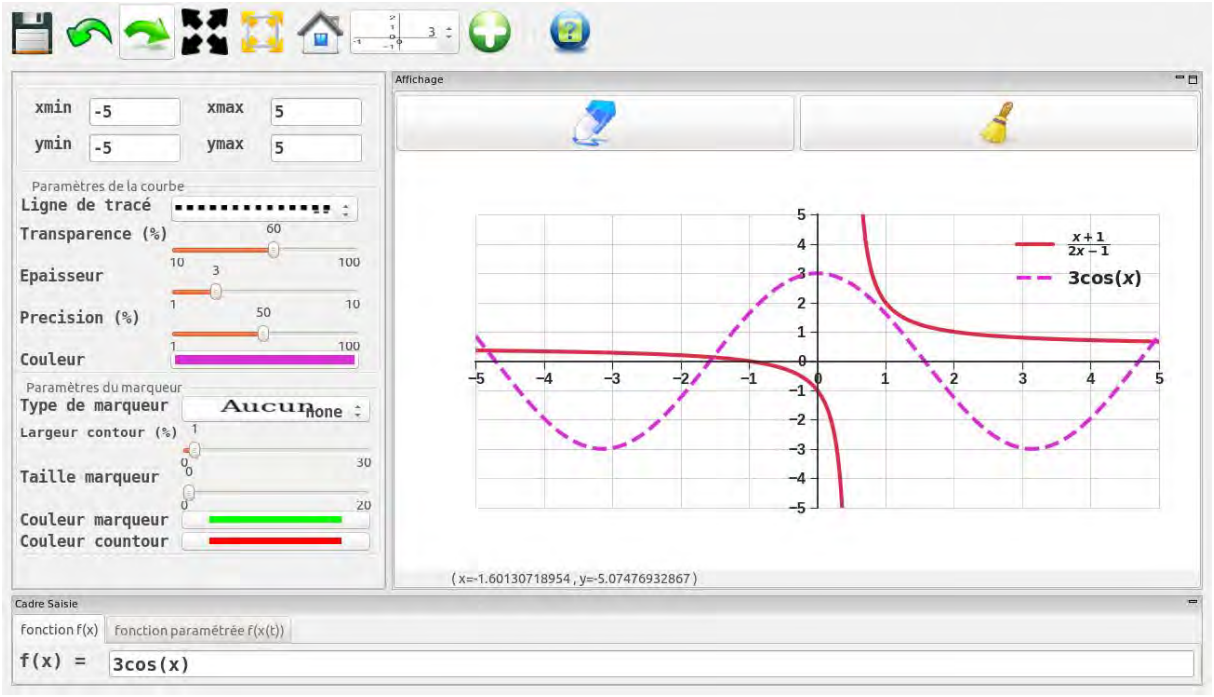


FIGURE 5.9 – SimulaMath et graphe d'une fonction

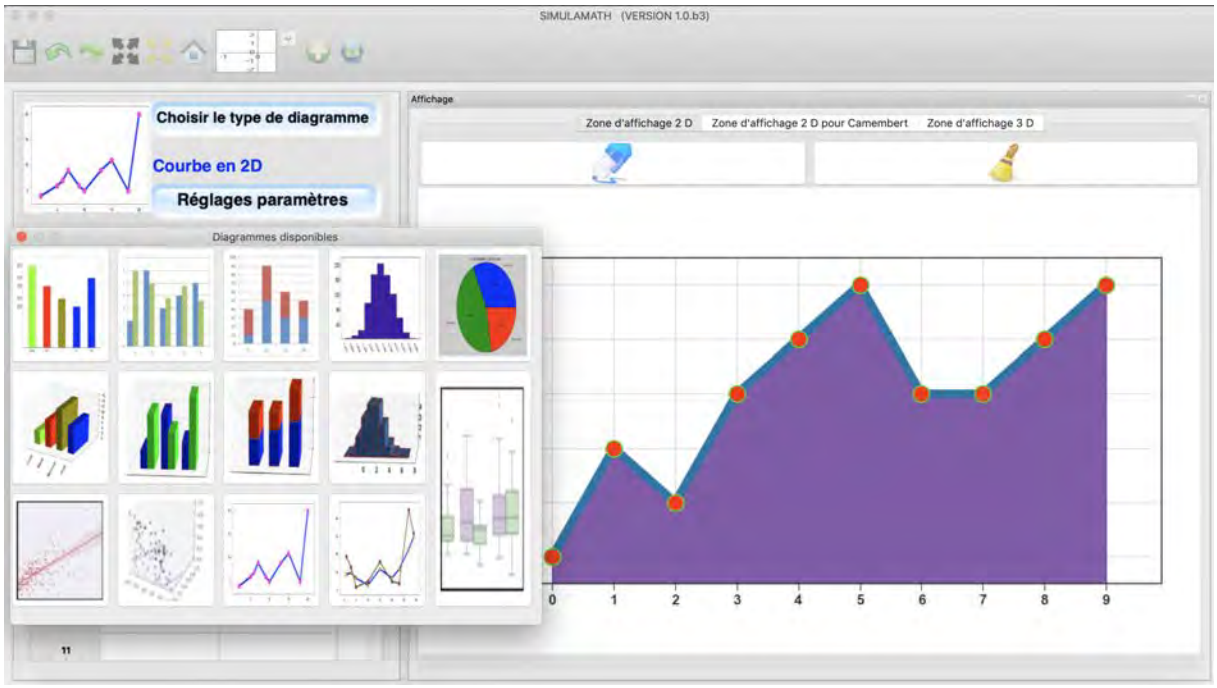


FIGURE 5.10 – SimulaMath et les courbes

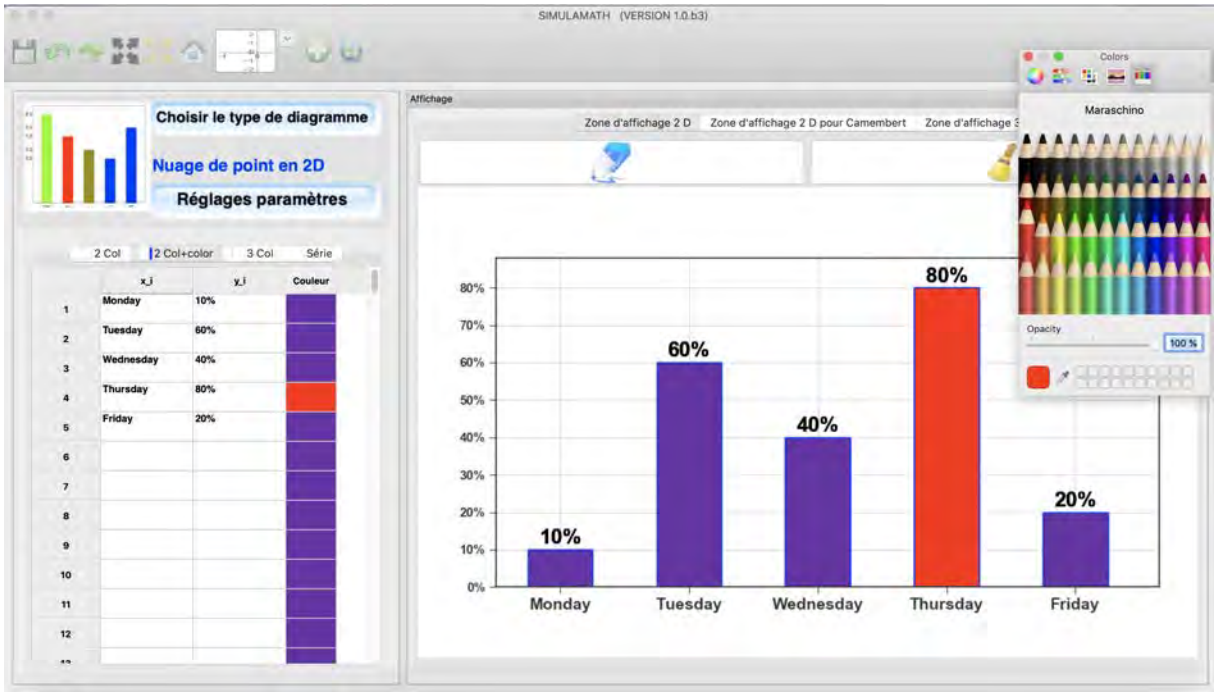


FIGURE 5.11 – SimulaMath et les diagrammes en barres

Notez que nous avons intégré des réseaux arithmétiques et que l'un de nos objectifs de les ajouter à SimulaMath est de faciliter la simulation des problèmes difficiles comme le problème de vecteur le plus court, le problème de vecteur le plus proche, le problème des minima successifs, etc.

5.8 SimulaMath et les probabilités

Les distributions de probabilités sont très importantes en cryptographie. Pour les schémas de chiffrement (par exemple dans la cryptographie basée les réseaux arithmétiques), les éléments sont choisis dans une distribution comme la distribution uniforme, binomiale ou gaussienne. Supposons qu'une variable aléatoire X suit la distribution normale de moyenne 0 et de variance 1, on peut facilement calculer une probabilité et faire une simulation comme suit.

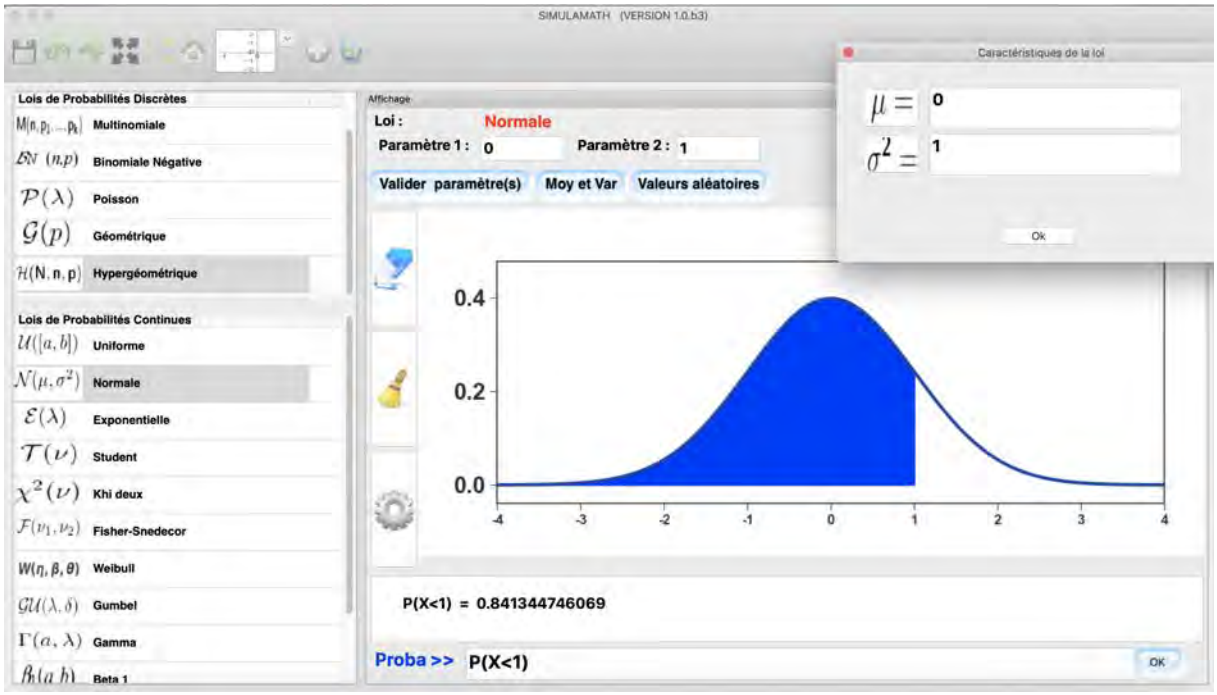


FIGURE 5.12 – SimulaMath et les distributions de probabilités

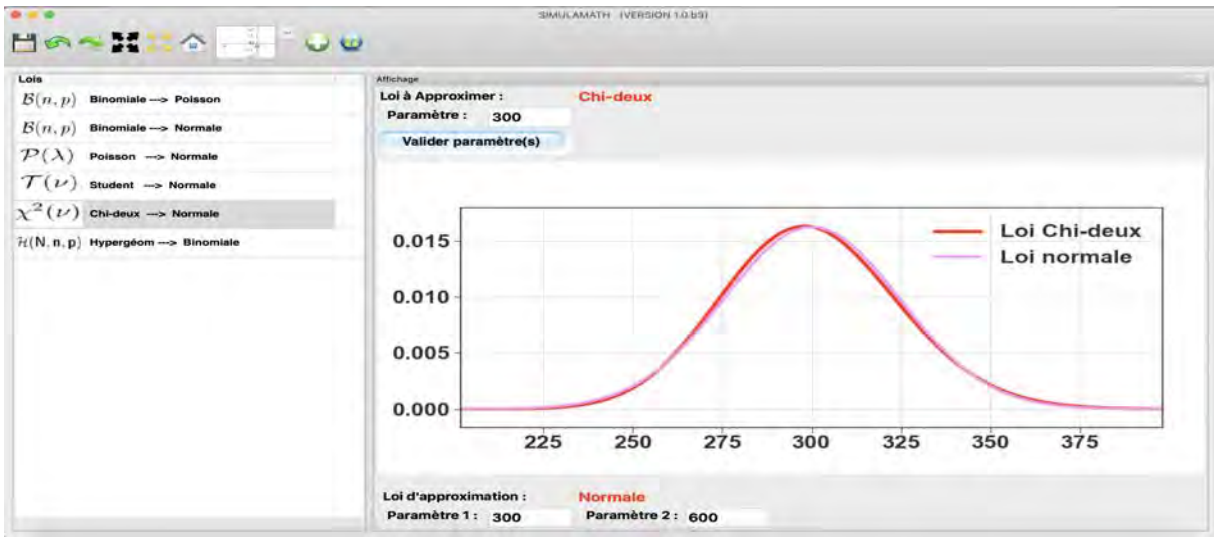


FIGURE 5.13 – SimulaMath et convergence des distributions de probabilités

5.9 SimulaMath et autres domaines

Nous avons décrit ci-dessus quelques-unes des applications du logiciel SimulaMath. Mais il y a d'autres domaines par exemple :

- ☛ la théorie des nombres
- ☛ les réseaux arithmétiques

- ☛ les bases de Groebner
- ☛ la statistique descriptive
- ☛ les tableurs

5.10 Conclusion

SimulaMath a atteint bon nombre de ses premiers objectifs ; ses objectifs actuels comprennent l'amélioration des performances pour la géométrie interactive, l'utilisation du module mlab de Mayavi [RV11] pour les graphiques à trois dimensions, l'intégration de la version anglaise du logiciel et l'extension de la communauté des développeurs. Nous prévoyons également d'ajouter dans les prochaines versions quelques techniques d'intelligence artificielle au logiciel et l'ajout d'un interpréteur qui sera propre au logiciel comme SageMath qui sera certainement une modification de l'interpréteur de Python afin de permettre la programmation sous SimulaMath.

Conclusion générale et Perspectives de recherche

Conclusion

Beaucoup de schémas et protocoles cryptographiques sur les courbes (hyper)elliptiques nécessitent le hachage ou la génération de points aléatoires sur une courbe ou la Jacobienne associée à une courbe. Plusieurs travaux sur la construction de fonctions d'encodage et de hachage ont été effectués, durant ces deux dernières décennies, sur les courbes dont la plupart ont été fait sur les courbes elliptiques.

Dans cette thèse, nous avons proposé plusieurs fonctions d'encodage et de hachage sur les familles de courbes hyperelliptiques $\mathbb{H}_2 : y^2 = f_2(x) = x^5 + a_3x^3 + a_1x + a_0$, $\mathbb{H}_3 : y^2 = f_3(x) = x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$, $\mathbb{H}_4 : y^2 = f_4(x) = x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$ et $\mathbb{H}_5 : y^2 = f_5(x) = x^{11} + a_9x^9 + a_7x^7 + a_5x^5 + a_3x^3 + a_1x + a_0$. Nous avons étendu notre approche à la forme courte de Weierstrass $\mathbb{H}_1 : y^2 = f_1(x) = x^3 + a_1x + a_0$. Nous avons trouvé une formule unifiant ces encodages en encodant sur la courbe hyperelliptique de genre $g \leqslant 9$, $\mathbb{H}_g : y^2 = f_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0$. Nos encodages présentent des propriétés intéressantes telles que la presque-injectivité et ils peuvent tous être inversés. Nous avons aussi résumé les différentes techniques les plus utilisées pour la construction de fonctions d'encodage et de hachage sur les courbes elliptiques et hyperelliptiques. Nous avons enfin, dans cette partie, effectué une étude comparative des différentes approches d'encodage et de hachage.

Nous avons également, dans cette thèse, décrit SimulaMath, un logiciel de calcul et de simulation pour l'apprentissage, l'enseignement et la recherche en mathématiques que nous avons conçu et développé avec le langage Python. Nous avons effectué une petite comparaison avec certains logiciels tels que SageMath et Geogebra. Nous avons également montré quelques cas d'utilisation du logiciel dans la résolution de problèmes dans certaines branches des mathématiques telles que les courbes elliptiques, les codes correcteurs, la théorie des nombres, l'algèbre linéaire, les probabilités etc.

Perspectives de recherche

Les fonctions d'encodage ont eu récemment des applications en cryptographie post-quantique notamment sur certains schémas basés sur le calcul d'isogénies. On peut citer, entre autres, les travaux de Bernstein *et al.* dans «Quantum Circuits for the CSIDH : Optimizing Quantum Evaluation of Isogenies» [BLMP19]) et ceux de Onuki *et al.* où ces derniers ont utilisé la fonction d'encodage Elligator[BLMP19] dans leur proposition d'implémentation en temps constant de CSIDH [OAYT19]. Il serait donc important de pousser davantage les recherches sur comment l'utilisation des fonctions d'encodage et de hachage pouvait être bénéfique dans la construction de schémas post-quantiques basés sur

les isogénies.

Même si beaucoup de travaux ont été faits sur la construction d'encodage et de fonctions de hachage sur les courbes elliptiques, il n'y a pas encore, à notre connaissance, une construction générale explicite sur la forme générale de Weierstrass sur un corps fini $\mathbb{F}_q$ sans conditions sur q et/ou les coefficients de la courbe. Il serait important d'aborder ce point dans le futur.

Pour les courbes hyperelliptiques, peu de constructions d'encodage ont été proposées, il serait aussi important de proposer des fonctions d'encodage et de hachage sur d'autres familles. On pourrait aussi essayer de généraliser notre fonction d'encodage générale ψ_g en genre $g \in \{1, 2, \ldots, 9\}$ à n'importe quel genre

Avec la menace de la machine quantique, les constructions actuelles de schémas sur les courbes sont basées, le plus souvent, sur les isogénies. À l'avenir, Il serait aussi intéressant de voir si les techniques utilisées dans la construction d'encodage et de fonctions de hachage pouvaient être adaptées à la construction d'isogénies explicites entre courbes elliptiques ou entre Jacobiennes de courbes hyperelliptiques.

Publications

① Michel Seck, Hortense Boudjou, Nafissatou Diarra and Ahmed Youssef Ould Cheikh, "On Indifferentiable Hashing into the Jacobian of Hyperelliptic Curves of Genus 2", Springer Lecture Notes in Computer Science (LNCS), In M. Joye and A. Nitaj, editors, Advances in Cryptology AFRICACRYPT-2017, 205-222(2017)

② Seck M., Diarra N. : Unified Formulas for Some Deterministic Almost-Injective Encodings into Hyperelliptic Curves. In : Joux A., Nitaj A., Rachidi T. (eds) Progress in Cryptology - AFRICACRYPT 2018. AFRICACRYPT 2018. Lecture Notes in Computer Science, vol 10831. Springer, Cham (2018)

③ Diarra N., Seck M. and Sow D.. A Note on Encoding and Hashing into Elliptic and Hyperelliptic Curves . In A Collection of Papers in Mathematics and Related Sciences, a festschrift in honour of the late Galaye Dia (Editors : Seydi H., Lo G.S. and Diakhaby A.). Spas Editions, Euclid Series Book, pp. 565 –593. (2018) Doi : 10.16929/sbs/2018.100-07-01

Preprints

① Seck, M. : SimulaMath : A software for teaching and research in mathematics. https://simulamath.org (2018)

Troisième partie

Annexes

Annexe A

Généralités sur la théorie algébrique des nombres

Contents

A.1 Groupes et Anneaux

A.1.1 Groupes

Définition A.1.1 (groupe)**.**

Un groupe G est un ensemble non vide muni d'une loi de composition interne $*$ telle que

1. $*$ est associative, c'est à dire que pour tous $x, y, z \in G$, on a $(x * y) * z = x * (y * z)$
2. $*$ admet un élément neutre : il existe $e \in G$ tel que pour tout $x \in G$, on a $x * e = e * x = x$
3. pour tout $x \in G$, il existe y (appelé symétrique de x) tel que $x * y = y * x = e$.

Remarque A.1.2

- *Le groupe G est dit commutatif ou abélien, si la loi de composition est commutative.*
- *La loi d'un* ***groupe additif*** *est souvent notée par $+$ ou $\dot{+}$ et l'élément neutre par 0 et dans le cas d'un* ***groupe multiplicatif*** *la loi est souvent notée $\times$ ou . et l'élément neutre par 1.*
- *Le symétrique de x est généralement désigné par x^{-1} (si le groupe est noté multiplicativement, on parle dans ce cas d'inverse) ou $-x$ (si le groupe est noté additivement, on parle dans ce cas d'opposé) .*

Définition A.1.3 (sous-groupe)**.**

Soit $(G, *)$ un groupe. Un sous-groupe H de G est un sous-ensemble de G contenant l'élément neutre e et tel que

- pour tous $x, y \in H$, on a $x * y \in H$.
- si $x \in H$ alors son symétrique est aussi dans H.

Exemple A.1.1 *Soit $(G, .)$ un groupe (noté multiplicativement) et soit $x \in G$. L'ensemble $\{x^n : n \in \mathbb{Z}\}$ est le sous-groupe de G engendré par x. Il est noté par $< x >$.*

Définition A.1.4 (ordre)**.**

— Le cardinal d'un groupe G, noté $\#G$ ou $|G|$, est aussi appelé son **ordre**. Le groupe G est fini si son ordre est fini.
— Soit G un groupe. Un élément $x \in G$ est d'ordre fini si $\prec x \succ$ est fini. Dans ce cas, l'ordre de x est $\# \prec x \succ$, c'est-à-dire le plus petit entier positif n tel que
  — $x^n = e$ si G est un groupe multiplicatif.
  — $nx = e$ si G est un groupe additif.
  Sinon, x est d'ordre infini.

Définition A.1.5 (groupe cyclique)**.**

Un groupe G est dit cyclique s'il est fini et s'il existe $x \in G$ tel que $\prec x \succ = G$. Si un tel élément x existe, on l'appelle un générateur de G.

Théorème A.1.6 (Lagrange)
Soit G un groupe fini et H un sous-groupe de G. Alors l'ordre de H divise l'ordre de G. Par conséquent, l'ordre de chaque élément divise également l'ordre de G.

Définition A.1.7 (homomorphisme de groupe)**.**

Soient G_1 et G_2 deux groupes avec les lois respectives $*$ et $\dot{+}$ et les éléments neutres e_1 et e_2. Un homomorphisme de groupe $\phi : G_1 \to G_2$ est une application de G_1 à G_2 telle que pour tout $x, y \in G_1, \phi(x * y) = \phi(x) \dot{+} \phi(y)$

Définition A.1.8 (action d'un groupe)**.**

Soit S un ensemble et $(G, *)$ un groupe. Le groupe G agit sur S s'il y a une application $\sigma : G \times S \to S$ telle que
— $\sigma(e, s) = s$ pour tout $s \in S$
— $\sigma(x, \sigma(y, s)) = \sigma(x * y, s)$ pour tout $s \in S$ et pour tous $x, y \in G$.

Définition A.1.9 (image et noyau d'un groupe)**.**

Soit $\phi : G_1 \to G_2$ un homomorphisme de groupe et soit e_{G_2} l'élément neutre de G_2. L'image (ou le rang) de ϕ est $\mathrm{Im}(\phi) = \{\phi(x) : x \in G_1\}$ et le noyau de ϕ est $\ker(\phi) = \{x \in G_1 : \phi(x) = e_{G_2}\}$

Il faut noter que $\mathrm{Im}(\phi) = \phi(G_1)$ et $\ker(\phi) = \phi^{-1}(e_{G_2})$ sont des sous-groupes de G_2 et G_1 respectivement.

A.1.2 Anneaux

Définition A.1.10 (anneau)**.**

Un anneau A est un ensemble non vide muni de deux lois de composition $+$ et $.$ telles que
- $(A, +)$ est un groupe abélien ;
- $.$ est associative, c'est-à-dire que $(x.y).z = x.(y.z)$ pour tous $x, y, z \in A$;
- $.$ est distributive par rapport à $+$, c'est-à-dire pour tous $x, y, z \in A, x.(y + z) = x.y + x.z$ et $(y + z).x = y.x + z.x$.

- On dit qu'un anneau A est unitaire s'il admet un élément neutre par rapport à la multiplication, noté 1_A ou simplement 1 (différent de 0 l'élément neutre de $+$).
- L'anneau A est dit commutatif, si la loi $\times$ est commutative, c'est-à-dire pour tous $x, y \in A, xy = yx$.

Dans la suite, sauf mention contraire, les anneaux sont unitaires et commutatifs dont l'identité (par rapport à .) sera notée par 1.

Définition A.1.11 (diviseur, unité)**.**

Soient A un anneau et $x, y, u \in A$.

1. y divise x (ou x est divisible par y), noté $y|x$, s'il existe $z \in A$ tel que $x = yz$.
2. x est un diviseur de zéro s'il existe $z \in A\backslash\{0\}$ tel que $xz = 0$.
3. u est unité s'il existe $u' \in A$ tel que $uu' = 1$. Et u' est l'inverse multiplicatif de u, noté $u' = u^{-1}$. Notez que l'ensemble des unités de A forme un groupe abélien noté A^*.

Exemple A.1.2 *L'ensemble des polynômes $\mathbb{Z}[x]$ à coefficients dans $\mathbb{Z}$ muni de l'addition et de la multiplication des polynômes est un anneau.*

Définition A.1.12 (homomorphisme d'anneaux)**.**

Soient $(A_1, +, .)$ et $(A_2, \dot{+}, \odot)$ deux anneaux. Un homomorphisme d'anneaux ψ est une application de A_1 à A_2 telle que pour tous $x, y \in A_1$, on a :
- $\psi(x + y) = \psi(x) \dot{+} \psi(y)$
- $\psi(x.y) = \psi(x) \odot \psi(y)$
- $\psi(1_{A_1}) = 1_{A_2}$.

Définition A.1.13 (anneau intègre)**.**

Un anneau A tel que pour tous $x, y \in A$, l'égalité $xy = 0$ implique que $x = 0$ ou $y = 0$ est appelé un anneau intègre.

Définition A.1.14 (idéal)**.**

Un sous-ensemble non vide I de l'anneau A est appelé idéal si
- I est un sous-groupe de A par rapport à la loi $+$,
- Pour tout $a \in A$ et pour tout $b \in I$, on a $ab \in I$.

Définition A.1.15 (idéal premier, idéal maximal, idéaux co-premiers)**.**

Soit $I \subsetneq A$ un idéal de A.
- I est premier si pour tous $x, y \in A$ avec $xy \in I$ alors $x \in I$ ou $y \in I$.
- I est maximal si pour tout idéal J de A l'inclusion $I \subset J$ entraine que $J = I$ ou $J = A$.
- Deux idéaux I et J de A sont co-premiers si $I + J = \{i + j : i \in I \text{ et } j \in J\}$ est égal à A.

Exemple A.1.3 *L'idéal* $< x^2 + 1 >$ *est maximal dans* $\mathbb{R}[x]$.

Remarque A.1.16
1. *Tout idéal maximal est également premier mais l'inverse n'est pas vrai en général.*
2. *Tout anneau commutatif unitaire contient un idéal maximal.*

Définition A.1.17 (anneau local)**.**

Un anneau A ayant un seul idéal maximal est appelé un anneau local.

Exemple A.1.4

Définition A.1.18 (idéal principal, anneau noethérien)**.**

Soit I un idéal d'un anneau A.

1. I est finiment engendré s'il existe des éléments $a_1, \ldots, a_n \in A$ tels que chaque $x \in I$ peut être écrit sous la forme $x = x_1a_1 + \ldots + x_na_n$ avec $x_1, \ldots, x_n \in A$.
2. I est principal s'il existe $a \in A$ tel que $I = aA$. A est un anneau principal si tout idéal de A est principal.
3. A est noethérien si chaque idéal I de A est finiment engendré.

Théorème A.1.19 (Théorème du reste chinois)
Soient $I_1, \ldots, I_k$ *des idéaux deux à deux co-premiers de* A*. Alors*

$$A/\prod_{i=1}^{k} I_i \simeq \prod_{i=1}^{k} A/I_i$$

Exemple A.1.5 *Pour tout entier* $n = p_1^{\alpha_1} \times p_2^{\alpha_2} \times \ldots \times p_k^{\alpha_k}$ *où* $p_1, \ldots, p_k$ *sont des nombres premiers, nous avons*

$$\frac{\mathbb{Z}}{n\mathbb{Z}} = \frac{\mathbb{Z}}{p_1^{\alpha_1}\mathbb{Z}} \times \frac{\mathbb{Z}}{p_1^{\alpha_2}\mathbb{Z}} \times \ldots \times \frac{\mathbb{Z}}{p_1^{\alpha_k}\mathbb{Z}}$$

Définition A.1.20 (anneau de division)**.**

Un anneau de division (pas nécessairement commutatif) est un anneau dans lequel chaque élément non nul est unité (inversible).

Notez que les anneaux de division commutatifs sont des corps (les corps sont définis dans la section A.2).

Théorème A.1.21 (Wedderburn, 1905)
Tout anneau de division fini est un corps

Preuve : Voir [Gri07].

□

A.1.3 Localisation

Définition A.1.22 (partie multiplicative)**.**

Soit A un anneau. Un sous-ensemble S de A est dit multiplicatif si $1 \in S$ et $ab \in S$ pour tous $a, b \in S$.

Considérons une partie multiplicative S d'un anneau A. Soit $\sim$ la relation d'équivalence définie sur $A \times S$ par :

$$(a_1, s_1) \sim (b_2, s_2) \Longleftrightarrow \exists t \in S : t(a_1 s_2 - a_2 s_1) = 0$$

La classe d'équivalence d'une paire $(a, s) \in A \times S$ est notée par $\frac{a}{s}$

Proposition A.1.23
Alors l'ensemble de toutes les classes d'équivalence de A

$$S^{-1}A = \frac{A \times S}{\sim} = \left\{\frac{a}{s} : a \in A, s \in S\right\}$$

est un anneau, muni des opérations $\frac{a}{s} + \frac{a'}{s'} = \frac{as' + a's}{ss'}$ *et* $\frac{a}{s}.\frac{a'}{s'} = \frac{aa'}{ss'}$

Définition A.1.24 (localisation)**.**

L'anneau $S^{-1}A$ est appelé le localisé de A par rapport à S.

Exemple A.1.6
1. *Si* $S = \{1_A\}$ *alors* $S^{-1}A = A$.
2. *Si* $S = \mathbb{Z}\backslash\{0\}$ *et* $A = \mathbb{Z}$, *alors* $S^{-1}A = \mathbb{Q}$.

A.2 Corps et Espaces vectoriels

A.2.1 Corps

Définition A.2.25 (corps).

Un corps K est un anneau commutatif tel que tout élément non nul est inversible.

Exemple A.2.7 *Si A est un anneau intègre, alors $\left(A\backslash\{0\}\right)^{-1}A$ est un corps appelé corps de fractions de A.*

Proposition A.2.26
Soient A un anneau et I un idéal de A. Alors l'ensemble quotient A/I est un corps si et seulement si I est maximal.

Définition A.2.27 (homomorphisme de corps).

Soient K et L des corps. Un homomorphisme de corps est un homomorphisme d'anneaux entre K et L.

Proposition A.2.28
Tout homomorphisme de corps est injectif.

Définition A.2.29 (caractéristique d'un corps).

Soit K un corps et soit ψ l'homomorphisme naturel de $\mathbb{Z}$ à K défini par

$$\psi(n) = \begin{cases} 1+1+\ldots+1 & \text{n fois si } n>0 \\ -(1+1+\ldots+1) & \text{-n fois sinon.} \end{cases}$$

Le noyau de ψ est de la forme $m\mathbb{Z}$, pour un entier positif m, qui est appelé la caractéristique de K et est noté par $\mathrm{char}(K)$.

Proposition A.2.30
La caractéristique d'un corps K est soit 0, soit un nombre premier p.

Proposition A.2.31
Tout sous-groupe multiplicatif fini d'un corps est cyclique.

Notez qu'un tel sous-groupe est constitué de racines de l'unité, puisque ses éléments sont d'ordre fini.

A.2.2 Espaces vectoriels

Définition A.2.32 (espace vectoriel)**.**

Un espace vectoriel V sur un corps K est un groupe abélien pour une première opération notée $+$, muni d'une multiplication par un scalaire de $K \times V$ dans V, qui associe (λ, x) à λx et tel que pour tous $x, y \in V$, pour tous $\lambda, \mu \in K$, on a

- $\lambda(x + y) = \lambda x + \lambda y$
- $(\lambda + \mu)x = \lambda x + \mu x$
- $(\lambda\mu)x = \lambda(\mu x)$
- $1x = x$.

Un élément x de V est un vecteur alors qu'un élément λ de K est appelé un scalaire.

Définition A.2.33 (bases d'un espace vectoriel)**.**

Une base K d'un espace vectoriel V est un sous-ensemble $B \subset V$ qui

- est linéairement indépendant sur K, c'est-à-dire que pour tout sous-ensemble fini $\{x_1, \ldots, x_n\} \subset B$ et tout $\lambda_1, \ldots, \lambda_n \in K$, $\sum_{i=1}^{n} \lambda_i x_i = 0 \Longrightarrow \lambda_i = 0 \ \forall i$
- engendre V sur K, c'est-à-dire que pour tout $x \in V$, il existe des vecteurs $x_1, \ldots, x_n$ et des scalaires $\lambda_1, \ldots, \lambda_n$ tels que $x = \sum_{i=1}^{n} \lambda_i x_i$.

Théorème A.2.34
Soit V un espace vectoriel sur K. Si V est différent de $\{0\}$ alors V possède une base.

Définition A.2.35 (dimension d'un espace vectoriel)**.**

Deux bases d'un espace vectoriel V sur K ont le même cardinal. Cet invariant est appelé la dimension sur K de V ou simplement la dimension de V. Notez que la dimension peut être infinie.

A.3 Extension de corps

A.3.1 Définitions et propriétés

Définition A.3.36 (sous-corps)**.**

Un sous-corps d'un corps K est un sous-ensemble F de K tel que F est un sous-groupe additif de K et $L\backslash\{0\}$ est un sous-groupe multiplicatif de $K\backslash\{0\}$.

Exemple A.3.8 *$\mathbb{Q}$ est un sous-corps de $\mathbb{R}$ et $\mathbb{R}$ est un sous-corps de $\mathbb{C}$.*

Proposition A.3.37
Tout corps K admet un plus petit sous-corps, qui est isomorphe à $\mathbb{Q}$ si K est de caractéristique 0, à $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ si K est de caractéristique $p \neq 0$.

Preuve : Voir [Gri07].

□

Définition A.3.38 (extension de corps)**.**

Soient K et L deux corps. On dit que L est une extension de K, et on note L/K, si K est un sous corps de L.

Définition A.3.39 (degré d'extension)**.**

Soit L/K une extension de corps. Alors L peut être considéré comme un K-espace vectoriel. La dimension de L/K est appelée le degré de L/K et est notée par $[L : K]$ ou $\deg(L/K)$. Si le degré de L/K est fini, alors on dira que l'extension L/K est finie.

Exemple A.3.9 *$\mathbb{C}$ est une extension finie de $\mathbb{R}$, avec $[\mathbb{C} : \mathbb{R}] = 2$, mais $\mathbb{R}$ est une extension infinie de $\mathbb{Q}$.*

Proposition A.3.40
Soient L/K et M/L deux extensions. Alors $[M : K] = [M : L][L : K]$

Définition A.3.41 (éléments algébriques et transcendantaux)**.**

Soit F/K une extension de corps. Un élément α de L est algébrique sur K quand $f(\alpha) = 0$ pour un polynôme non nul $f(X) \in K[X]$. Sinon, α est transcendantal sur K.

Exemple A.3.10 *Chaque nombre complexe est algébrique sur $\mathbb{R}$. Les nombres réels e et π sont transcendants sur $\mathbb{Q}$.*

Proposition A.3.42 ([Gri07])
α est algébrique sur K si et seulement si $[K(\alpha) : K]$ est fini.

Définition A.3.43 (degré d'un élément algébrique)**.**

Soit α un nombre algébrique sur K. Le polynôme irréductible unitaire $q \in K[X]$ (de degré minimal noté souvent $\min_K(\alpha)$) tel que $q(\alpha) = 0$ est le polynôme minimal de α sur K ; le degré de α sur K est le degré de $\min_K(\alpha)$.

Exemple A.3.11 $\min_{\mathbb{R}}(i) = X^2 + 1$ *et* $\min_{\mathbb{Q}}(\sqrt{2}) = X^2 - 2$.

Proposition A.3.44
Soient K un corps et $q \in K[X]$ un polynôme irréductible. A isomorphisme près, $L = K[X]/(q)$ est une simple extension de K, i.e $L = K(\alpha)$, où $\alpha = X + (q)$. De plus, $[L : K] = \deg q$ et $q = \min_K(\alpha)$.

Preuve : Voir Grillet [Gri07]

□

A.3.2 Extensions algébriques et normales, Corps de nombres

Définition A.3.45 (extension algébrique-transcendante)**.**

Une extension de corps L/K est algébrique, et L est algébrique sur K, lorsque chaque élément de L est algébrique sur K. Une extension de corps L/K est transcendante, et L est transcendant sur K, lorsque certains éléments de L sont transcendants dans K.

Exemple A.3.12 $\mathbb{C}$ *est une extension algébrique de* $\mathbb{R}$ *et* $\mathbb{R}$ *est une extension transcendante de* $\mathbb{Q}$.

Notez que chaque corps K admet une extension algébrique qui contient une racine de chaque polynôme non constant à coefficients dans K.

Proposition A.3.46
Toute extension de corps fini est algébrique.

Preuve : Voir Grillet [Gri07].

□

Théorème A.3.47 ([Gri07])
Tout corps K admet une extension algébrique $\bar{K}$ qui est algébriquement close. De plus, $\bar{K}$ est unique à K-isomorphisme près.

Définition A.3.48 (clôture algébrique)**.**

Une clôture algébrique d'un corps K est une extension algébrique $\bar{K}$ de K qui est algébriquement close.

Définition A.3.49 (extension normale)**.**

Une extension normale d'un corps K est une extension algébrique L de K telle que tout polynôme irréductible $q \in K[X]$ ayant une racine dans L est un produit de facteurs linéaires dans $L[X]$ (q admet toutes ses racines dans L).

Définition A.3.50 (corps de nombres)**.**

Un corps de nombres K est une extension algébrique de $\mathbb{Q}$ de degré fini. Un élément de K est appelé un nombre algébrique.

Soit A un anneau intègre, et soit K un corps contenant A.

Définition A.3.51 (élément entier)**.**

Un élément α de K est dit entier sur A s'il est racine d'un polynôme unitaire de $A[x]$, c'est-à-dire qu'il existe $a_0, \dots a_{n-1} \in A$ tel que $\alpha^n + a_{n-1}\alpha^{n-1} + \dots a_1\alpha + a_0 = 0$

Théorème A.3.52
L'ensemble des éléments de K entiers sur A forme un anneau.

Définition A.3.53 (anneau des entiers)**.**

L'anneau des éléments entiers sur A est appelé fermeture intégrale de A dans K. La fermeture intégrale de $\mathbb{Z}$ dans un corps de nombres algébriques K est appelée l'anneau des entiers $\mathcal{O}_K$ dans K. De plus, K est le corps de fractions de $\mathcal{O}_K$.

Définition A.3.54 (entier algébrique)**.**

Un nombre complexe qui est entier sur $\mathbb{Z}$ (qui est racine d'un polynôme à coefficients dans $\mathbb{Z}$) est appelé entier algébrique.

Exemple A.3.13 *$\sqrt{3}$ est un entier algébrique car il annule le polynôme $x^2 - 3$.*

Définition A.3.55 (domaine intégralement clos)**.**

A est intégralement clos s'il est lui-même intégralement clos dans son corps de fractions K, c'est-à-dire que si $\alpha \in K$ est entier sur A alors $\alpha \in A$.

A.3.3 Extensions séparables et inséparables

Définition A.3.56 (élément-extension séparable)**.**

Un élément α est séparable sur K si α est algébrique sur K et $\min_K(\alpha)$ est séparable (n'admet pas de racines multiples). Une extension algébrique L de K est séparable, et L est séparable sur K, lorsque chaque élément de L est séparable sur K.

Proposition A.3.57
Si un corps K est de caractéristique 0, alors chaque extension algébrique de K est séparable.

Définition A.3.58 (extension purement inséparable)**.**

Une extension algébrique L/K est purement inséparable, et L est purement inséparable sur K, lorsqu'aucun élément de L/K n'est séparable sur K.

Proposition A.3.59
Pour chaque extension algébrique L de K, l'ensemble $S = \{\alpha \in L : \alpha$ est séparable sur $K\}$ est un sous-corps de L, S est séparable sur K, et L est purement inséparable sur S.

A.3.4 Extension galoisienne

Définition A.3.60 (extension galoisienne)**.**

Une extension de Galois (ou galoisienne) d'un corps K est une extension normale et séparable L de K ; et on dira que L est Galois dans K.

Notez que si la caractéristique de K est 0, alors chaque extension normale de K est une extension de Galois de K. Un corps fini de caractéristique p est une extension de Galois de $\mathbb{Z}_p$.

Définition A.3.61 (groupe de Galois)**.**

Soit L/K une extension de corps. Le groupe de Galois $\mathrm{Gal}(L/K)$ d'une extension galoisienne L d'un corps K, aussi appelé le groupe de Galois de L sur K, est le groupe de tous les K-automorphismes de L laissant K invariant.

Proposition A.3.62 ([Gri07])
Si L est une extension de Galois de K, alors $\mathrm{Gal}(L/K) = [L : K]$

Définition A.3.63.

Soit L un corps et soit G un groupe d'automorphismes de L. Le corps laissé invariant de G est

$$\mathrm{Fix}_L(G) = \{x \in L : \sigma(x) = x \text{ pour tout } \sigma \in G\}$$

Notez que $\mathrm{Fix}_L(G)$ est un sous-corps de L.

Exemple A.3.14 $\mathrm{Fix}_{\mathbb{C}}(G) = \mathbb{R}$ *si* $G = \mathrm{Gal}(\mathbb{C}/\mathbb{R})$.

Proposition A.3.64

1. *Si G est un groupe fini d'automorphismes d'un corps L, alors L est une extension galoisienne finie de $F = \mathrm{Fix}_L(G)$ et $\mathrm{Gal}(F/L) = G$.*
2. *Si L est une extension galoisienne de K, alors le corps laissé invariant par $\mathrm{Gal}(L/K)$ est K.*

Preuve : Voir [Gri07].

□

Théorème A.3.65 ([Gri07])
Soit L une extension galoisienne finie d'un corps K.

1. *Si F est un sous-corps de L qui contient K, alors L est une extension finie de Galois de F et F est le corps laissé invariant par $\mathrm{Gal}(L/F)$.*
2. *Si H est un sous-groupe de $\mathrm{Gal}(L/K)$, alors $F = \mathrm{Fix}_L(H)$ est un sous-corps de L qui contient K, et $\mathrm{Gal}(L/F) = H$.*

Proposition A.3.66 ([Gri07])
Tout groupe abélien fini est le groupe de Galois d'une extension de $\mathbb{Q}$.

A.3.5 Norme et Trace

Pour une transformation linéaire T d'un espace vectoriel de dimension finie V, le déterminant (**resp.** la trace) de T est défini comme le déterminant (**resp.** la trace : la somme de toutes les entrées diagonales) de la matrice de T dans toute base de V. Une extension finie L d'un corps K est un espace vectoriel fini sur K , et la multiplication par λ est une transformation linéaire $x : \lambda x$ de L.

Définition A.3.67 (norme et trace)**.**

Soit L une extension finie d'un corps K. La norme $N_{L/K}(\alpha)$ et la trace $\mathrm{Tr}_{L/K(\alpha)}$ de $\alpha \in L$ sur K sont le déterminant et la trace de la transformation linéaire $T_\alpha : x \to \alpha x$ de L.

Notez que $N_{L/K}(\alpha)$ et $\mathrm{Tr}_{L/K}(\alpha)$ sont tous des éléments de K.

Proposition A.3.68
Soit L/K une extension finie de degré n. Soient $\alpha_1, \ldots, \alpha_r \in \bar{K}$ les conjugués distincts de $\alpha \in L$. Alors r divise n et

$$N_{L/K}(\alpha) = (\alpha_1 \times \alpha_2 \ldots \times \alpha_r)^{n/r} \text{ et } \mathrm{Tr}_{L/K}(\alpha) = \frac{n}{r}(\alpha_1 + \alpha_2 \ldots + \alpha_r)$$

Proposition A.3.69
Soit L/K une extension finie et $\alpha \in L$.

1. *Si $\alpha \in K$, alors $N_{L/K}(\alpha) = \alpha^n$ et $\mathrm{Tr}_{L/K}(\alpha) = n\alpha$ où $n = [L : K]$;*

2. *si $L = K(\alpha)$ est séparable sur K, alors $N_{L/K}(\alpha)$ est le produit des conjugués de α, et $\mathrm{Tr}_{L/K}(\alpha)$ est leur somme ;*
3. *Si L n'est pas séparable sur K, alors $\mathrm{Tr}_{L/K}(\alpha) = 0$;*
4. *Si L est une extension Galoisienne sur K, avec comme groupe de Galois G, alors*

$$N_{L/K}(\alpha) = \prod_{\sigma \in G} \sigma(\alpha) \text{ et } \mathrm{Tr}_{L/K}(\alpha) = \sum_{\sigma \in G} \sigma(\alpha)$$

Preuve : Découle de la proposition précédente

□

Nous avons les propriétés suivantes de la norme et de la trace.

Proposition A.3.70
Soit L/K une extension finie. Alors
— $N_{L/K}(\alpha + \beta) = N_{L/K}(\alpha) + N_{L/K}(\beta)$ *pour tous* $\alpha, \beta \in L$.
— $\mathrm{Tr}_{L/K}(\alpha.\beta) = \mathrm{Tr}_{L/K}(\alpha).\mathrm{Tr}_{L/K}(\beta)$ *pour tous* $\alpha, \beta \in L$.

Preuve : Voir [Gri07].

□

A.4 Extensions cyclotomiques

Définition A.4.71 (racine n-ième de l'unité)**.**

Les racines n-ième de l'unité dans $\mathbb{C}$ sont les nombres complexes

$$\varepsilon_k = \exp(2\pi ki/n), \text{ pour } 0 \leqslant k < n$$

Notez que la racine n-ième de l'unité ε_k est primitive si et seulement si k et n sont premiers entre eux.

Définition A.4.72 (polynôme cyclotomique)**.**

Le polynôme cyclotomique d'ordre n $\Phi_n(X) \in \mathbb{C}[X]$ est le produit de tous les $X - \varepsilon_k$ où ε_k est une racine n-ième primitive de l'unité, i.e

$$\Phi_n(X) = \prod_k (X - \varepsilon_k)$$

Exemple A.4.15 $\Phi_1(X) = X - 1, \Phi_2(X) = X + 1$ *et* $\Phi_4(X) = X^2 + 1$

Proposition A.4.73 ([Gri07])
Pour tous entiers n et $q \geqslant 2$, $\Phi_n(q) \in \mathbb{R}$ et $\Phi_n(q) > q - 1$.

Proposition A.4.74 ([Gri07])

$$X^n - 1 = \prod_{d/n} \Phi_d(X)$$

Proposition A.4.75 ([Gri07])
Pout tout entier $n > 0$,

1. $\Phi_n(X) \in \mathbb{Z}[X]$
2. $\Phi_n(X)$ *est irréductible dans* $\mathbb{Q}[X]$.

Définition A.4.76 (corps cyclotomique)**.**

Le corps cyclotomique d'ordre n est $\mathbb{Q}(\varepsilon_n) \subseteq \mathbb{C}$ où $\varepsilon_n \in \mathbb{C}$ est une nième racine primitive de l'unité.

Proposition A.4.77 ([Gri07])
Le corps $\mathbb{Q}(\varepsilon_n)$ est une extension de Galois de $\mathbb{Q}$; $[\mathbb{Q}(\varepsilon_n) : \mathbb{Q}] = \phi(n)$ où ϕ est la fonction d'Euler ; et $\mathrm{Gal}(Q(\varepsilon_n)/\mathbb{Q})$ est isomorphe au groupe des unités U_n de $\mathbb{Z}_n$.

A.5 Caractères sur des groupes abéliens finis

Définition A.5.78 (caractère)**.**

Soit $(G, *)$ un groupe abélien fini. Un caractère sur G est un homomorphisme de groupe χ de G dans $\mathbb{C}*$ où $\mathbb{C}*$ est le groupe multiplicatif des nombres complexes non nuls, c'est-à-dire $\chi(g_1 * g_2) = \chi(g_1)\chi(g_2)$ et $\chi(e) = 1$ avec e l'élément neutre de G.

Exemple A.5.16 *Le caractère trivial ε défini par $\varepsilon(g) = 1$ pour tout $g \in G$.*

L'ensemble des caractères sur G, muni de la multiplication $(\chi_1\chi_2)(g) = \chi_1(g)\chi_2(g)$ et $\chi^{-1}(g) = \chi(g^{-1})$ est un groupe abélien appelé groupe de caractères de G, et est noté par $\bar{G}$.

Théorème A.5.79
Soit G un groupe abélien fini. Il existe alors un isomorphisme de G dans $\bar{G}$. En particulier, $|\bar{G}| = |G|$

Théorème A.5.80 (Relations d'orthogonalité)
Soit G un groupe abélien (noté multiplicativement) d'ordre n (d'élément neutre 1) avec le groupe de caractères $\bar{G}$. Soit 1_G le caractère trivial de G.

1. *pour tout $\chi \in \bar{G}$, on a* $\displaystyle\sum_{a \in G} \chi(a) = \begin{cases} n & si\ \chi = 1_G \\ 0 & si\ \chi \neq 1_G \end{cases}$
2. *pour tout $a \in G$, on a* $\displaystyle\sum_{\chi \in \bar{G}} \chi(a) = \begin{cases} n & si\ a = 1 \\ 0 & si\ a \neq\neq 1 \end{cases}$

A.5.1 Résidus quadratiques et symboles de Legendre et de Jacobi

Soit $\mathbb{F}_q$ un corps fini à q éléments, q une puissance d'un nombre premier impair p.

Définition A.5.81 (résidu quadratique)**.**

On dit qu'un entier a est un résidu quadratique modulo p si l'équation $x^2 \equiv a(\bmod p)$ possède une solution.

Remarque A.5.82

- 0 *et* 1 *sont toujours des résidus quadratiques modulo* p*.*
- *Si* $a \equiv 0(\bmod p)$*, alors* $x = 0$ *est la seule solution.*
- *On suppose que* $a \neq 0(\bmod p)$*. Si l'équation* $x^2 \equiv a(\bmod p)$ *admet une solution, alors elle a deux solutions.*
- *Il y a* $(p-1)/2$ *résidus quadratiques (non nuls) dans* $\mathbb{F}_p$ *modulo* p *et le même nombre de non-résidus quadratiques (non nuls).*

Définition A.5.83 (symbole de Legendre)**.**

Le symbole de Legendre $\left(\frac{a}{p}\right)$ est défini par

$$\left(\frac{a}{p}\right) = \begin{cases} -1 & \text{si } a \text{ n'est pas un résidu quadratique } \bmod p \\ 0 & \text{si } a \equiv 0(\bmod p) \\ 1 & \text{si } a \text{ est un résidus quadratique mod } p \end{cases}$$

Nous avons les propriétés suivantes du symbole Legendre.

Théorème A.5.84

1. $\left(\frac{a}{p}\right) = a^{(p-1)/2}(\bmod p)$
2. $\left(\frac{ab}{p}\right) = \left(\frac{a}{p}\right)\left(\frac{b}{p}\right)$
3. $\left(\frac{a}{p}\right) = \left(\frac{b}{p}\right)$ *si* $a \equiv b(\bmod p)$
4. $\left(\frac{-1}{p}\right) = (-1)^{(p-1)/2}$
5. $\left(\frac{2}{p}\right) = (-1)^{(p^2-1)/8}$*, en particulier si* $p \equiv 1$ *ou* $7 \bmod 8$*,* $\left(\frac{2}{p}\right) = 1$*.*
6. *Si* p *et* q *sont deux nombres premiers impairs, alors on a la loi de* ***réciprocité quadratique*** *suivante :*

$$\left(\frac{q}{p}\right)\left(\frac{p}{q}\right) = (-1)^{(p-1)(q-1)/4}$$

Le symbole de Legendre peut être étendu dans le cas où p n'est pas nécessairement un nombre premier.

Définition A.5.85 (symbole de Kronecker–Jacobi)**.**

Soient $a, b \in \mathbb{Z}$, où b est positif et impair, et $b = \prod_i p_i^{r_i}$ où p_i est un nombre premier. Le symbole de Kronecker-Jacobi $\left(\frac{a}{b}\right)$ est défini par :

$$\left(\frac{a}{b}\right) = \prod_i \left(\frac{a}{p_i}\right)^{r_i}$$

A.5.2 Caractères sur les corps finis

Pour un corps fini $\mathbb{F}_q$, $q = p^n$ une puissance d'un nombre premier p, on peut considérer le groupe additif $(\mathbb{F}_q, +)$ ou le groupe multiplicatif $(\mathbb{F}_q^*, .)$.

Définition A.5.86 (caractère additif-multiplicatif)**.**

— Un caractère additif sur $\mathbb{F}_q$ est un caractère $\psi : (\mathbb{F}_q, +) \to \mathbb{C}^*$.
— Un caractère multiplicatif sur $\mathbb{F}_q$ est un caractère $\chi : (\mathbb{F}_q^*, .) \to \mathbb{C}^*$. On peut étendre χ à $\mathbb{F}_q$ comme suit :

$$\chi(0) = \begin{cases} 1 & \text{si } \chi = \varepsilon \text{ le caractère trivial} \\ 0 & \text{si } \chi \neq \varepsilon \end{cases}$$

Proposition A.5.87

1. *Les caractères multiplicatifs de $\mathbb{F}_q$ sont donnés par :*

$$\forall x \in \mathbb{F}_q, \chi(x) = e_q(x) = \exp(2\pi i x/q) \in \mathbb{C}^*$$

2. *Les caractères additifs de $\mathbb{F}_q$ sont donnés par*

$$\psi(x) = e_q(\mathrm{Tr}(x))$$

où $\mathrm{Tr}(x)$ est la trace de x.

Définition A.5.88 (somme de Gauss)**.**

Soit χ un caractère multiplicatif et ψ un caractère additif sur $\mathbb{F}_q$. La somme de Gauss associée à χ et ψ est la quantité

$$G(\chi, \psi) = \sum_{x \in \mathbb{F}_q} \chi(x)\psi(x)$$

1. Si ψ est trivial mais pas χ, alors $G(\chi, \psi) = 0$.
2. Si χ est trivial mais pas ψ, alors $G(\chi, \psi) = -1$.
3. Si les deux (χ et ψ) sont triviaux, alors $G(\chi, \psi) = q - 1$.

Définition A.5.89 (somme de Jacobi)**.**

Soit χ et ν deux caractères multiplicatifs sur $\mathbb{F}_q$. La somme de Jacobi associée à χ et ν est définie par

$$J(\chi, \psi) = \sum_{x+y=1} \chi(x)\nu(x) \text{ avec } x, y \in \mathbb{F}_q^*$$

Remarque A.5.90
$J(\chi, \psi) = q$ si les deux caractères χ et ν sont triviaux.

A.6 Racine carrée et carré dans un corps fini

Soit q une puissance positive d'un nombre premier impair p, c'est-à-dire, $q = p^n$, avec $n \geqslant 1$ et soit $\mathbb{F}_q$ le corps fini à q éléments. Le problème de calcul d'une racine carrée d'un élément arbitraire $a \in \mathbb{F}_q$ consiste à trouver un second élément $b \in \mathbb{F}_q$ tel que $b^2 = a$. Nous savons que le nombre d'éléments $a \in \mathbb{F}_q^*$ tels qu'un tel $b \in \mathbb{F}_q$ existe est exactement $(q-1)/2$.

Proposition A.6.91 (test de résiduosité quadratique)
On définit $\chi_q : a \mapsto a^{(q-1)/2}, a \in \mathbb{F}_q$. La racine carrée d'un élément $a \in \mathbb{F}_q^*$ existe si et seulement si $\chi_q(a) = 1$.

Deux techniques classiques et non-déterministes pour calculer les racines carrées dans les extensions de corps sont les algorithmes de Tonelli-Shanks et de Cipolla-Lehmer [AR14].

L'algorithme suivant due à Shank [Sha72] calcule une racine carrée d'un élément $a \in \mathbb{F}_q$, si elle existe quand $q \equiv 3(\bmod 4)$.

Algorithme 16 : L'algorithme de Shanks pour $q \equiv 3(\bmod 4)$

Entrées : $a \in \mathbb{F}_q^*$
Sortie : x (s'il existe) satisfaisant $x^2 = a$ ou $\perp$ sinon.

$a_1 := a^{(q-3)/4}$;
$a_0 := a_1(a_1 a)$;
Si $a_0 = -1$ **Alors**
  Retourner $\perp$
Fin Si
$x := a_1 a$.
Retourner x

Quand $q \equiv 5(\bmod 8)$, Atkin [Atk92] donne un algorithme efficace pour calculer une

racine carrée en $\mathbb{F}_q$.

Algorithme 17 : L'algorithme d'Atkin pour $q \equiv 5(\bmod 8)$

Entrées : $a \in \mathbb{F}_q^*$
Sortie : x (s'il existe) satisfaisant $x^2 = a$ ou $\perp$ sinon
/* PRECOMPUTATION */
$t := 2^{(q-5)/8}$;
/* COMPUTATION */
$a_1 := a^{(q-5)/8}$;
$a_0 := (a_1^2 a)^2$;
Si $a_0 = -1$ **Alors**
  Retourner $\perp$
Fin Si
$b : ta_1$;
$i := 2(ab)b$;
$x := (ab)(i-1)$.
Retourner x

Nous décrivons l'algorithme de Tonelli-Shanks [Ton91, Sha72] qui est un algorithme probabiliste dont la plupart des méthodes d'extraction de racines carrées sont dérivées,

comme l'algorithme [AR14] de Cipolla-Lehmer.

Algorithme 18 : Algorithme de Tonelli-Shanks

Entrées : $a \in \mathbb{F}_q^*$
Sortie : x (s'il existe) satisfaisant $x^2 = a$ ou $\perp$ sinon
`/* Legendre */`
Écrire $q - 1 = 2^s t$, où t est impair ;
$c_0 := 1$;
Tant que $c_0 = 1$ **Faire**
  Choisir aléatoirement $c \in \mathbb{F}_q^*$;
  $z := c^t,\ c_0 = c^{2^{s-1}}$;
Fin
`/* COMPUTATION */`
$w := a^{(t-1)/2}$;
$a_0 := (w^2 a)^{2^{s-1}}$;
Si $a_0 = -1$ **Alors**
  Retourner $\perp$
Fin Si
$v := s,\ x := aw,\ b = xw.$
Tant que $b \neq 1$ **Faire**
  Trouver le plus petit entier $k \geqslant 0$ tel que $b^{2^k} = 1$;
  $w := z^{2^{v-k-1}},\ z := w^2,\ b := bz,\ x := xw,\ v := k.$
Fin
Retourner x

Annexe B

Généralités sur les variétés, les courbes elliptiques et hyperelliptiques

Contents

La plupart des notions abordées dans cette annexe sont extraites de [Har77, Sil09, MhWZ98, CF06, Eng01, Was03].

B.1 Variétés affines et projectives

Soientt $\mathbb{K}$ un corps et $\bar{\mathbb{K}}$ sa clôture algébrique. Dans cette section, $\bar{\mathbb{K}}[X]$ désigne l'anneau des polynômes à $n+1$ variables, c'est-à-dire $\bar{\mathbb{K}}[X] = \bar{\mathbb{K}}[X_0, X_1, \ldots, X_n]$.

B.1.1 Variétés affines

Définition B.1.1 (espace affine)**.**

Un espace affine de dimension n (sur $\bar{\mathbb{K}}$) est l'ensemble des n-uplet

$$\mathbb{A}^n = \mathbb{A}^n(\bar{\mathbb{K}}) = \{P = (x_1, \ldots, x_n) : x_i \in \bar{\mathbb{K}}\}$$

L'ensemble des points $\mathbb{K}$-rationnels de $\mathbb{A}^n$ noté $\mathbb{A}^n(\mathbb{K})$ est l'ensemble

$$\mathbb{A}^n(\mathbb{K}) = \{P = (x_1, \ldots, x_n) \in \mathbb{A}^n : x_i \in \mathbb{K}\}$$

Un élément $P = (x_1, \ldots, x_n) \in \mathbb{A}^n$ est appelé un **point**.

Remarque B.1.2
Plus généralement, pour tout corps $\mathbb{K}'$ tel que $\mathbb{K} \subset \mathbb{K}' \subset \bar{\mathbb{K}}$, l'ensemble des points $\mathbb{K}'$-rationnels de $\mathbb{A}^n$ noté $\mathbb{A}^n(\mathbb{K}')$ est l'ensemble $\mathbb{A}^n(\mathbb{K}') = \{P = (x_1, \ldots, x_n) \in \mathbb{A}^n : x_i \in \mathbb{K}'\}$

Soit $\bar{\mathbb{K}}[X] = \bar{\mathbb{K}}[X_1, \ldots, X_n]$ l'anneau des polynômes à n variables, et soit $I \subset \bar{\mathbb{K}}[X]$ un idéal. On associe à I un sous-ensemble de $\mathbb{A}^n$,

$$V_I = \{P \in \mathbb{A}^n : f(P) = 0 \quad \forall f \in I\}$$

Définition B.1.3 (ensemble affine)**.**

Un ensemble algébrique (affine) est un ensemble de la forme V_I. Si V est un ensemble algébrique, l'idéal de V est donné par

$$I(V) = \{f \in \bar{\mathbb{K}}[X] : f(P) = 0 \quad \forall P \in V\}$$

Un ensemble algébrique V est défini sur $\mathbb{K}$, et on note $V/\mathbb{K}$, si son idéal $I(V)$ peut être généré par des polynômes dans $\mathbb{K}[X]$. Si V est défini sur $\mathbb{K}$, alors l'ensemble des points $\mathbb{K}$-rationnels de V est l'ensemble $V(\mathbb{K}) = V \cap \mathbb{A}^n(\mathbb{K})$.

Proposition B.1.4
L'union de deux ensembles algébriques est un ensemble algébrique. L'intersection de toute famille d'ensembles algébriques est un ensemble algébrique. L'ensemble vide et l'espace entier sont des ensembles algébriques.

Preuve : Voir [Har77]

□

Définition B.1.5 (topologie de Zariski)**.**

On définit la topologie de Zariski sur $\mathbb{A}^n$ en prenant les sous-ensembles ouverts comme les compléments des ensembles algébriques.

Définition B.1.6 (variété affine)**.**

Un ensemble algébrique affine V est appelé une variété **(affine)** si $I(V)$ est un idéal premier dans $\bar{\mathbb{K}}[X]$.

Notons que si V est défini sur $\mathbb{K}$, il n'est pas suffisant de vérifier que $I(V/\mathbb{K})$ est premier dans $\mathbb{K}[X]$.

Définition B.1.7 (anneau de coordonnées, corps de fonctions)**.**

- L'**anneau de coordonnées** d'une variété algébrique affine $V/\mathbb{K}$ définie sur $\mathbb{K}$ est l'anneau quotient $\mathbb{K}[V] = \mathbb{K}[X]/I(V/K)$
- Le corps de fractions de $\mathbb{K}[V]$ est noté $\mathbb{K}(V)$ et est appelé **le corps de fonctions** de $V/\mathbb{K}$. Il est à noter que si V est une variété, $I(V)$ est un idéal premier et que le quotient $\mathbb{K}[X]/I(V)$ est un domaine intégral, ce qui signifie que $\mathbb{K}[V]$ admet un corps de fractions.

Notons que, on peut définir de façon similaire $\bar{\mathbb{K}}[V]$ et $\bar{\mathbb{K}}(V)$ en remplaçant $\mathbb{K}$ par $\bar{\mathbb{K}}$.

Définition B.1.8 (dimension d'une variété)**.**

Soit V une variété. La dimension de V, notée $\dim(V)$, est le degré de transcendance de $\bar{\mathbb{K}}(V)$ sur $\bar{\mathbb{K}}$.

La dimension de $\mathbb{A}^n$ est n et si $V \subset \mathbb{A}^n$ est défini par une seule équation polynomiale non constante $f(X_1, \ldots X_n) = 0$ alors $\dim V = n - 1$.

Définition B.1.9 (variété lisse)**.**

Soient V une variété, $P \in V$, et $f_1, \ldots, f_m \in \bar{\mathbb{K}}[X]$ un ensemble générant $I(V)$. Alors V est **lisse** (ou non singulière) en P si la matrice $m \times n$ $\left(\frac{\partial f_i}{\partial X_j}(P)\right)_{\substack{1 \leqslant i \leqslant m \\ 1 \leqslant j \leqslant n}}$ est de rang $n - \dim(V)$. Si V est non singulière en chaque point, alors on dit que V est **lisse** (ou non singulière).

B.1.2 Variété projective

Définition B.1.10 (espace projectif)**.**

L'espace projectif de dimension n sur $\bar{\mathbb{K}}$, noté $\mathbb{P}^n$ ou $\mathbb{P}^n(\bar{\mathbb{K}})$, est l'ensemble

$$\mathbb{P}^n = \frac{\mathbb{A}^{n+1}(\bar{\mathbb{K}}) \backslash \{0\}}{\sim}$$

où $\sim$ est la relation d'équivalence définie par : $(X_0, \ldots X_n) \sim (Y_0, \ldots Y_n)$ s'il existe un $\lambda \in \bar{\mathbb{K}}^*$ tel que $X_i = \lambda Y_i$ pour tout i. Une classe d'équivalence $\{(\lambda X_0, \ldots, \lambda X_n) : \lambda \in \bar{\mathbb{K}}^*\}$ dénotée par $[X_0, \ldots, X_n]$ est appelée un point projectif ; et $X_0, \ldots X_n$ sont appelés coordonnées (homogènes) pour le point correspondant dans $\mathbb{P}^n$.
L'ensemble des points $\mathbb{K}$-rationnels dans $\mathbb{P}^n$ est l'ensemble

$$\mathbb{P}^n(K) = \{[X_0, \ldots X_n] \in \mathbb{P}^n : X_i \in \mathbb{K}\}$$

Définition B.1.11 (polynôme homogène, idéal homogène)**.**

Un polynôme $f \in \bar{\mathbb{K}}[X]$ est dit homogène de degré d si

$$f(\lambda X_0, \lambda X_1, \ldots, \lambda X_n) = \lambda^d f(X_0, X_1, \ldots, X_n) \quad \forall \lambda \in \bar{\mathbb{K}}$$

Un idéal $I \subset \bar{\mathbb{K}}[X]$ est homogène s'il est généré par des polynômes homogènes.

A chaque idéal homogène I on associe un sous-ensemble de $\mathbb{P}^n$ par

$$V_I = \{P \in \mathbb{P}^n : f(P) = 0 \text{ pour tout polynôme homogène } f \in V\}$$

Définition B.1.12 (ensemble algébrique projectif)**.**

Un ensemble algébrique (projectif) est tout ensemble de la forme V_I pour un idéal I.

Si V est un ensemble algébrique projectif, l'idéal (homogène) de V, noté par $I(V)$, est l'idéal de $\bar{\mathbb{K}}[X]$ généré par

$$\{f \in \bar{\mathbb{K}}[X] : f \text{ est homogène et } f(P) = 0 \text{ pour tout } P \in V\}$$

Un tel V est défini sur $\mathbb{K}$, noté $V/\mathbb{K}$, si son idéal $I(V)$ peut être généré par des polynômes homogènes dans $\mathbb{K}[X]$. Si V est défini sur $\mathbb{K}$, alors l'ensemble des points $\mathbb{K}$-rationnels de V est l'ensemble $V(\mathbb{K}) = V \cap \mathbb{P}^n$.

Définition B.1.13 (variété projective)**.**

Un ensemble algébrique projectif est appelé une **variété projective** si son idéal homogène $I(V)$ est un idéal premier en $\bar{\mathbb{K}}[X]$.

Définition B.1.14 (dimension d'une variété projective)**.**

1. Soit $V/\mathbb{K}$ une variété projective. On choisit $\mathbb{A}^n \subset \mathbb{P}^n$ tel que $V \cap \mathbb{A}^n \neq \varnothing$. La dimension de V est la dimension de $V \cap \mathbb{A}^n$.
2. Le corps de fonction d'une variété projective V , noté $\mathbb{K}(V)$, est le corps de fonction de $V \cap \mathbb{A}^n$, et de même pour $\bar{\mathbb{K}}(V)$.

Proposition B.1.15
Une variété projective $V \subset \mathbb{P}^n$ est de dimension $n-1$ si et seulement si elle est l'ensemble des zéros d'un seul polynôme homogène irréductible f de degré positif. V est appelé une hypersurface dans $\mathbb{P}^n$.

Preuve : Voir [Har77].

□

B.1.3 Points à l'infini

On définit l'inclusion $\phi_i : \mathbb{A}^n \to \mathbb{P}^n : (y_1, \ldots, y_n) \mapsto [y_1, \ldots, y_{i-1}, 1, y_i, \ldots, y_n]$ for $1 \leqslant i \leqslant n$. Ce qui montre que $\mathbb{P}^n$ contient n copies de $\mathbb{A}^n$. On définit l'hyperplan dans $\mathbb{P}^n$ $H_i = \{[x_0, \ldots, x_n] : x_i = 0\}$ et son complément $U_i = \mathbb{P}^n \backslash H_i$. Alors l'application $\psi_i : \mathbb{A}^n \to U_i : (y_1, \ldots, y_n) \mapsto \phi_i(y_1, \ldots, y_n)$ est une bijection. Cette bijection permet d'identifier chaque sous-ensemble V de $\mathbb{A}^n$ à un sous-ensemble de $\mathbb{P}^n$.

Définition B.1.16 (homogénéisation, déshomogénéisation)**.**

— Pour tout $f(Y) \in \bar{\mathbb{K}}[Y]$, on définit

$$f^*(X_0, \ldots, X_n) = X_i^d f\left(\frac{X_0}{X_i}, \frac{X_1}{X_i}, \ldots, \frac{X_{i-1}}{X_i}, \frac{X_{i+1}}{X_i}, \ldots, \frac{X_n}{X_i}\right)$$

où $d = \deg(f)$ est le plus petit entier pour lequel f^* est un polynôme. On dit que f^* est l' **homogénéisation** de f par rapport à X_i.

— Le processus de remplacement du polynôme $f(X_0, \ldots, X_n)$ par le polynôme $f(Y_1, \ldots, Y_{i-1}, 1, Y_{i+1}, \ldots, Y_n)$ est appelé **déshomogénéisation** de f par rapport à X_i.

Définition B.1.17 (Clôture projective)**.**

Soit $V \subset \mathbb{A}^n$ un ensemble algébrique affine. La clôture projective de V , notée par $\bar{V}$, est l'ensemble algébrique projectif dont l'idéal homogène $I(\bar{V})$ est généré par l'ensemble $\{f^*(X) : f \in I(V)\}$.

Définition B.1.18 (point à l'infini)**.**

Soit $\bar{V}$ la clôture projective d'une variété affine V. Les points à l'infini sur V sont les points de $\bar{V} \backslash V$.

Exemple B.1.1 *1. Soient $a, b \in \mathbb{K}$ tel que $-16(27b^2+4a^3) \neq 0$. Pour la variété affine $V_1 = \{(x, y) \in \mathbb{A}^2(\mathbb{K}) : y^2 = x^3 + ax + b\}$, nous avons un seul point à l'infini qui est égal à $[0, 0, 1]$.*

2. Soient $a, b \in \mathbb{K}$ tels que $ab(a^2 - b^2) \neq 0$. Pour la variété affine

$$V_2 = \left\{(x, y) \in \mathbb{A}^2(\mathbb{K}) : ax(y^2 - 1) = by(x^2 - 1)\right\}$$

nous avons trois points à l'infini qui sont $[1, 0, 0], [0, 1, 0]$ et $[a, b, 0]$.

B.1.4 Applications entre variétés

Dans cette section, nous examinons les applications algébriques entre variétés projectives.

Définition B.1.19 (application rationnelle)**.**

Soient V_1 et V_2 des variétés projectives dans $\mathbb{P}^n$. On dit que $\phi : V_1 \to V_2$ est une application rationnelle s'il existe un ensemble de fonctions rationnelles $f_0, \ldots, f_n \in \bar{\mathbb{K}}(V_1)$ tel que pour tout $P \in V_1$, $\phi(P) = [f_0(P), \ldots, f_n(P)]$. Si, de plus, il y a quelques $\lambda \in \bar{\mathbb{K}}^*$ tels que $\lambda f_0, \ldots, \lambda f_n \in \mathbb{K}(V_1)$ alors ϕ est dite définie sur $\mathbb{K}$.

Définition B.1.20 (application rationnelle régulière)**.**

Une application rationnelle $\phi = [f_0, \ldots, f_n] : V_1 \to V_2$ est régulière (ou définie) en $P \in V_1$ s'il existe une fonction $g \in \bar{\mathbb{K}}(V_1)$ telle que chaque gf_i est régulière en P i.e il existe quelques i pour lesquels $(gf_i)(P) \neq 0$.

Définition B.1.21 (isomorphisme)**.**

Soient V_1 et V_2 des variétés. On dit que V_1 et V_2 sont isomorphes, et on écrit $V_1 \cong V_2$, s'il y a des morphismes $\phi : V_1 \to V_2$ et $\psi : V_2 \to V_1$ tels que $\phi \circ \psi$ et $\psi \circ \phi$ sont les applications identité sur V_1 et V_2 respectivement.

Notons que $V_1/\mathbb{K}$ et $V_2/\mathbb{K}$ sont isomorphes sur $\mathbb{K}$ si ϕ et ψ peuvent être définis sur $\mathbb{K}$.

B.2 Courbes algébriques

Dans cette section, nous présentons des propriétés de base sur les variétés projectives de dimension un, qui seront nécessaires pour notre étude sur les courbes elliptiques et hyperelliptiques. Par courbe, nous entendons toujours une variété projective de dimension un. Nous allons nous focaliser particulièrement sur les courbes lisses lesquelles sont importantes dans la théorie des courbes elliptiques et hyperelliptiques.

B.2.1 Morphisme de courbes et ramification

Définition B.2.22 (fonction rationnelle)**.**

Soit P un point sur une courbe C. Une fonction rationnelle $r \in \bar{\mathbb{K}}(C)$ est dite régulière ou définie en P s'il existe $f, g \in \bar{\mathbb{K}}[C]$ telle que $r = \dfrac{f}{g}$ et $g(P) \neq 0$. La valeur de r à P est alors $\dfrac{f(P)}{g(P)}$. L'anneau de toutes les fonctions rationnelles régulières en P est appelé l'anneau local de C en P et est noté $\mathcal{O}_P(C)$. On écrit parfois $r(P) = \infty$ si r n'est pas régulière en P.

Définition B.2.23 (morphisme de courbes)**.**

Un morphisme de C_1 à C_2 est définie par une application rationnelle qui est régulière en tout point de C_1.

Définition B.2.24 (degré d'un morphisme)**.**

Soit $\phi : C_1 \to C_2$ un morphisme non constant de courbes sur $\mathbb{K}$. Le degré de ϕ est le degré de l'extension $[\mathbb{K}(C_1) : \phi^\star(\mathbb{K}(C_2))]$ où

$$\phi^\star : \mathbb{K}(C_2) \to \mathbb{K}(C_1) : f \mapsto f \circ \phi$$

Si un morphisme est constant, son degré est défini comme étant égal à 0.

Définition B.2.25 (paramètre d'uniformisation, ordre fonction rationnelle)**.**

Soit C une courbe et $P \in C$.

- Le paramètre d'uniformisation pour P est une fonction $u \in \mathcal{O}_P$ avec $u(P) = 0$ et vérifiant : pour chaque polynôme g avec $g(P) = 0$, il existe un entier unique d et une fonction s vérifiant $s(P) \notin \{0, \infty\}$ et $g = u^d s$.
- L'entier unique d vérifiant $g = u^d s$ est appelé l'ordre de g en P, noté $\mathrm{ord}_P(g)$.
- Si $r = \dfrac{f}{g} \in \bar{\mathbb{K}}(C)$, l'ordre de r en P est défini comme suit :

$$\mathrm{ord}_P(r) = \mathrm{ord}_P(f) - \mathrm{ord}_P(g)$$

Définition B.2.26 (ramification sauvage et modérée)**.**

Soit $\phi : C_1 \to C_2$ une application rationnelle non constante de courbes lisses, $P \in C_1$ et U un paramètre d'uniformisation pour $\phi(P)$. Alors le nombre

$$e_\phi(P) = \mathrm{ord}_P(U \circ \phi)$$

est appelé l'indice de ramification de ϕ en P.

- Si $e_\phi > 1$, ϕ est dite ramifiée en P, sinon elle est non ramifiée en P. ϕ est dite non ramifiée si elle est non ramifiée en tout point de C_1.
- Si $\mathrm{char}(\mathbb{K}) = 0$, ou si $\mathrm{char}(\mathbb{K}) = p$ et que p ne divise pas e_ϕ, on dit que la ramification est modérée (tame). Si p divise e_ϕ, elle est sauvage (wild).

Proposition B.2.27

Soit $\phi : C_1 \to C_2$ une application rationnelle non constante de courbes lisses.

1. *Pour tout $Q \in C_2$,*
$$\sum_{P \in \phi^{-1}(Q)} e_\phi(P) = \deg(\phi)$$
2. *Soit $\psi : C_2 \to C_3$ une autre application rationnelle non constante de courbes lisses. Alors pour tout $P \in C_1$,*
$$e_{\psi\circ\phi}(P) = e_\phi(P) e_\psi(\phi(P))$$
3. *ϕ n'est pas ramifiée si et seulement si $\#\phi^{-1}(Q) = \deg(\phi)$ pour tout $Q \in C_2$.*

Preuve : Voir [Sil09].

□

Proposition B.2.28
Soit $\phi : C_1 \to C_2$ un morphisme de courbes. Alors ϕ est soit constant, soit surjectif.

Proposition B.2.29
Soient C_1 et C_2 des courbes lisses, et soit $\phi : C_1 \to C_2$ un morphisme de degré un. Alors, ϕ est un isomorphisme.

B.2.2 Diviseurs et Jacobienne

Dans cette sous-section, nous supposons que C est une courbe algébrique.

Définition B.2.30 (diviseur, degré, ordre)**.**

— Un diviseur D est une somme formelle de points de C.
$D = \sum_{P \in C} n_P P, n_P \in \mathbb{Z}$ où seul un nombre fini des n_P est non nul.
— Le degré de D, noté $\deg D$, est l'entier $\sum_{P \in C} n_P$
— L'ordre de D en P est l'entier n_P, et on écrit $\mathrm{ord}_P(D) = n_P$
— Le support de D est l'ensemble $\mathrm{supp}(D) = \{P \in C : n_P \neq 0\}$.

L'ensemble des diviseurs sur C, noté $\mathbb{D}(C)$, forme un groupe additif selon la règle d'addition suivante : $\sum_{P \in C} m_P P + \sum_{P \in C} n_P P = \sum_{P \in C} (m_P + n_P) P$. L'ensemble de tous les diviseurs de degré 0, noté $\mathbb{D}^\circ(C)$, est un sous-groupe de $\mathbb{D}(C)$.

Définition B.2.31 (diviseur d'une fonction rationnelle)**.**

Soit $R \in \bar{\mathbb{K}}(C)^\star$. Le diviseur associé à R est défini par $\mathrm{div}(R) = \sum_{P \in C} \mathrm{ord}_P(R) P$.

Définition B.2.32 (diviseur principal)**.**

Un diviseur $D \in \mathbb{D}$ est principal s'il est de la forme $D = \mathrm{div}(R)$ pour un certain $R \in \bar{\mathbb{K}}(C)^*$.

Notons que l'application $\mathrm{div} : \bar{\mathbb{K}}(C)^* \to \mathbb{D}$ est un homomorphisme de groupes abéliens. Ainsi, l'ensemble des diviseurs principaux est un sous-groupe de $\mathbb{D}(C)$; il est noté par $\mathbb{P}\mathrm{rinc}(C)$.

Proposition B.2.33
Une fonction rationnelle a autant de zéros que de pôles, en comptant les multiplicités ; et $\deg \mathrm{div}(R) = 0$ *pour toute fonction rationnelle* $R \in \bar{\mathbb{K}}(C)^\star$.

Preuve : Voir [Sil09, MhWZ98].

□

Proposition B.2.34
Soit Δ *un diviseur principal. Alors la fonction rationnelle associée au diviseur* Δ *est unique à multiplication par une constante non nulle près dans* $\bar{\mathbb{K}}$.

Définition B.2.35 (groupe de Picard)**.**

- Deux diviseurs sont linéairement équivalents, et on écrit $D_1 \sim D_2$, si $D_1 - D_2$ est principal.
- Le groupe de Picard de C, noté $\mathrm{Pic}(C)$, est le quotient de $\mathbb{D}(C)$ par son sous-groupe des diviseurs principaux $\mathbb{P}\mathrm{rinc}(C)$. i.e $\mathrm{Pic}(C) = \dfrac{\mathbb{D}(C)}{\mathbb{P}\mathrm{rinc}(C)}$. Le groupe de Picard de degré 0 de C est $\mathrm{Pic}^0(C) = \dfrac{\mathbb{D}^\circ(C)}{\mathbb{P}\mathrm{rinc}(C)}$.

Définition B.2.36 (diviseur effectif)**.**

Un diviseur $D = \sum n_p P$ est effectif ou positif, noté $D \geqslant 0$, si $n_P \geqslant 0$ pour tout $P \in C$. De même, pour deux diviseurs quelconques $D_1, D_2 \in \mathbb{D}$, on écrit $D_1 \geqslant D_2$ si le diviseur $D_1 - D_2$ est effectif.

Notons que $\geqslant$ est un ordre partiel sur $\mathbb{D}$.

Définition B.2.37.

Soit $D \in \mathbb{D}$. Nous associons à D l'ensemble des fonctions

$$\mathcal{L}(D) = \{f \in \bar{\mathbb{K}}(C)^* : \mathrm{div}(f) - D \geqslant 0\} \bigcup \{0\}$$

L'ensemble $\mathcal{L}(D)$ est un espace vectoriel $\bar{\mathbb{K}}$ de dimension finie, et nous notons sa dimension par $l(D)$.

Théorème B.2.38 (Riemann–Roch)
*Soit C une courbe lisse.. Il existe un entier $g \geqslant 0$, appelé **genre** de C et $K_C \in \mathbb{D}(C)$ (appelé diviseur canonique) tels que pour tout diviseur $D \in \mathbb{D}(C)$, on a*

$$l(D) - l(K_C - D) = \deg D - g + 1$$

Preuve : Voir [Har77].

□

Nous rappelons ici les formules de Riemann Hurwitz qui mettent en relation la ramification d'un morphisme séparable avec son degré, ainsi que le genre des courbes

Définition B.2.39 (morphisme séparable et inséparable)**.**

Soit $\phi : C_1 \to C_2$ un morphisme. On dit que ϕ est séparable (inséparable ou purement inséparable) si l'extension de corps $\bar{\mathbb{K}}(C_1)/\phi^*(\bar{\mathbb{K}}(C_2))$ est séparable (inséparable ou purement inséparable).

Théorème B.2.40 (formule de Riemann–Hurwitz)
Soit $\phi : C_1 \to C_2$ un morphisme séparable non constant de courbes lisses. Soient g_{C_1} et g_{C_2} le genre de C_1 et C_2 respectivement. Alors nous avons

$$2g_{C_1} - 2 \geqslant (\deg \phi)(2g_{C_2} - 2) + \sum_{P \in C_1} (e_\phi(P) - 1)$$

avec égalité si et seulement si ϕ est modérément ramifié. $e_\phi(P)$ indique l'indice de ramification de ϕ en P.

Théorème B.2.41
Soit $\phi : C_1 \to C_2$ un morphisme fini de courbes sur un corps $\mathbb{K}$. Supposons que $\mathbb{K}(C_1)$ est une extension purement inséparable de $\mathbb{K}(C_2)$. Alors $g_{C_1} = g_{C_2}$.

Définition B.2.42 (groupe algébrique [CF06])**.**

Un groupe algébrique (absolument irréductible) $\mathcal{G}$ sur un corps $\mathbb{K}$ est une variété (affine ou projective) absolument irréductible définie sur $\mathbb{K}$ avec trois propriétés supplémentaires :

a) l'addition qui est un morphisme $m : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$

b) l'inverse qui est un morphisme $i : \mathcal{G} \to \mathcal{G}$

c) l'élément neutre qui est un point $\mathbb{K}$-rationnel $0 \in \mathcal{G}(\mathbb{K})$

satisfaisant les lois habituelles de groupe :

$$m \circ (Id_{\mathcal{G}} \times m) = m \circ (m \times Id_{\mathcal{G}}),\ m_{|\{0\}\times\mathcal{G}} = p_2 \text{ et } m \circ (i \times Id_{\mathcal{G}}) \times \delta_0 = c_0$$

où p_2 est la projection de $\mathcal{G} \times \mathcal{G}$ sur le second argument, δ_0 est l'application diagonale de $\mathcal{G}$ à $\mathcal{G} \times \mathcal{G}$ et c_0 est l'application qui envoie $\mathcal{G}$ à 0.

Remarquons que l'on peut noter $m(P, Q)$ par $P \oplus Q$ et $i(P)$ par $-P$ pour tout $P, Q \in \mathcal{G}(\mathbb{K})$.

Définition B.2.43 (variété abélienne)**.**

Une variété abélienne A est une variété projective lisse munie d'une structure de groupe algébrique.

Théorème B.2.44
Soit C une courbe lisse, projective et absolument irréductible de genre g sur un certain corps $\mathbb{K}$. Il existe alors une variété abélienne J de dimension g sur $\mathbb{K}$ tel que $J(\mathbb{K}) \cong \mathrm{Pic}^0(C(\mathbb{K}))$.

Définition B.2.45 (Jacobienne)**.**

La variété abélienne J est appelée la variété Jacobienne (ou simplement Jacobienne) de la courbe C.

Il faut noter que l'avantage de l'identification de la Jacobienne J avec le groupe Pic^0 est que l'on peut représenter les points sur J par des diviseurs sur C. On peut aussi utiliser cette représentation pour faire des calculs dans le groupe $J(\mathbb{K})$.

Théorème B.2.46 (Mordell-Weil)
Soit $\mathbb{K}$ un corps de nombre et soit J la Jacobienne d'une courbe C sur $\mathbb{K}$. Alors le groupe $J(\mathbb{K})$ est un groupe abélien ***finiment généré****.*

Ce théorème a été prouvé par Mordell en 1922 pour les courbes elliptiques sur le corps des rationnels $\mathbb{Q}$ et généralisé par Weil quelques années plus tard.

B.3 Courbes elliptiques

B.3.1 Définitions et propriétés de base

Définition B.3.47 (variété absolument irréductible [CF06]).

Une variété V de l'espace affine $\mathbb{A}^n$ (projective $\mathbb{P}^n$) sur $\mathbb{K}$ est dite absolument irréductible si elle est irréductible comme ensemble fermé par rapport à la topologie de Zariski des espaces correspondants sur $\bar{\mathbb{K}}$.

Une courbe elliptique sur $\mathbb{K}$ est une **courbe projective non singulière absolument irréductible** définie sur $\mathbb{K}$ de **genre 1** avec au moins un point $\mathbb{K}$-rationnel.

Définition B.3.48 (équation de Weierstrass).

Une courbe elliptique sur $\bar{\mathbb{K}}$ est une clôture projective d'une courbe non singulière définie par une **équation de Weierstrass**

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{B.1}$$

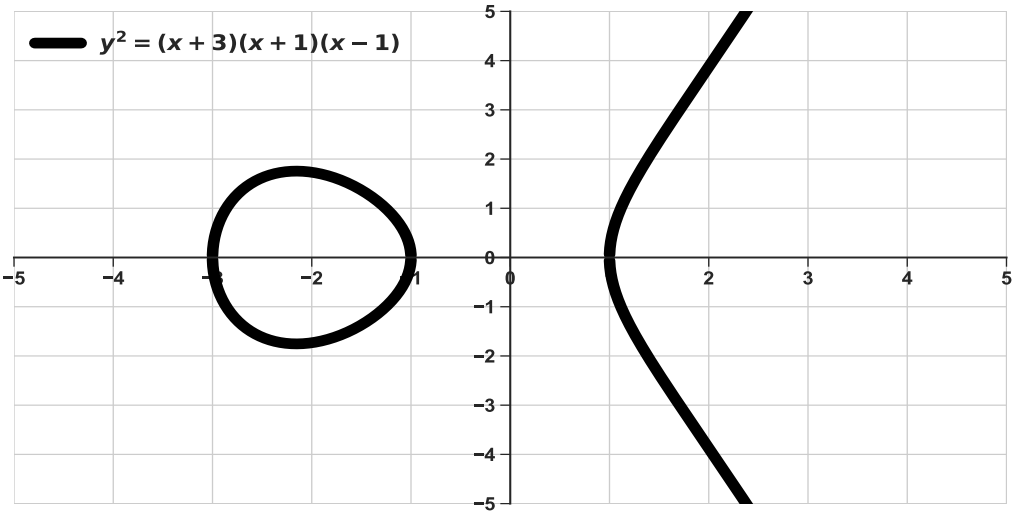
avec $a_1, a_2, a_3, a_4, a_6 \in \bar{\mathbb{K}}$. Si $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$, alors on dit que E est définie sur $\mathbb{K}$ et on note $E/\mathbb{K}$.

Remarque B.3.49
Notons que l'équation projective de Weierstrass de E est une équation de la forme

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

Il n'a qu'un seul point à l'infini $\mathcal{O} = [0, 1, 0]$.

FIGURE B.1 – Courbe elliptique : $y^2 = (x + 3)(x + 1)(x - 1)$ sur $\mathbb{R}$ (réalisé avec SimulaMath [1])



On définit les quantités suivantes :

$$
\begin{aligned}
&b_2 = a_1^2 + 4a_4, \qquad b_4 = 2a_4 + a_1a_3, \qquad b_6 = a_3^2 + 4a_6.\\
&b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2,\\
&c_4 = b_2^2 - 24b_4, \qquad c_6 = -b_2^2 + 36b_2b_4 - 216b_6,\\
&\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6,\\
&j = c_4^3/\Delta
\end{aligned}
$$

Définition B.3.50 (discriminant, j-invariant)**.**

La quantité Δ est appelée le discriminant de la courbe elliptique E et le nombre j est le j-invariant.

Théorème B.3.51 (courbes elliptiques isomorphes)
Deux courbes elliptiques définies par les équations de Weierstrass

$$
\begin{aligned}
E &: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\\
E' &: y^2 + a_1'xy + a_3'y = x^3 + a_2'x^2 + a_4'x + a_6'
\end{aligned}
$$

sont isomorphes si E' peut être obtenu à partir de E par un changement de variables de la forme

$$
\psi : \begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} u^2x + r \\ u^3y + u^2sx + t \end{pmatrix} = \begin{pmatrix} u^2 & 0 \\ u^2s & u^3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} r \\ t \end{pmatrix}
$$

où $u \in \bar{\mathbb{K}}^*, r, s, t \in \bar{\mathbb{K}}$ (et en divisant l'équation ainsi obtenue par u^6). La transformation correspondante ψ est considérée comme un changement admissible de variables.

Preuve : Voir Enge [Eng01].

□

Remarque B.3.52
Notons que les changements de variables admissibles représentent les seuls isomorphismes de la catégorie des sous-variétés algébriques du plan affine qui conservent la forme de l'équation de Weierstrass.

Exemple B.3.2 *Soit E une courbe elliptique sur $\bar{\mathbb{K}}$ définie par*

$$
E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6
$$

Si $\mathrm{Char}(\bar{\mathbb{K}}) \neq 2$*, les changements admissibles des variables $(x, y) \mapsto (x, y - \frac{1}{2}(a_1x + a_3))$ transforment E en une courbe elliptique*

$$
E : y^2 = x^3 + a_2'x^2 + a_4'x + a_6'
$$

1. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

Si de plus, $\text{Char}(\bar{\mathbb{K}}) \neq 3$*, un processus similaire peut être appliqué à droite pour éliminer le terme* x^2*. Les changements admissibles des variables* $(x, y) \mapsto (x - \frac{1}{3}a_2', y)$ *transforment* E' *en une courbe*

$$E'' : y^2 = x^3 + a_4''x + a_6''$$

Proposition B.3.53

1. *L'équation de Weierstrass* (B.1) *est non singulière si et seulement si son discriminant* $\Delta \neq 0$*.*
2. *Deux courbes elliptiques* E_1 *et* E_2 *sont isomorphes sur* $\bar{\mathbb{K}}$ *si et seulement si elles ont le même j-invariant, c'est-à-dire* $j(E_1) = j(E_2)$*.*
3. *Soit* $j_0 \in \bar{\mathbb{K}}$*. Il existe une courbe elliptique définie sur* $\mathbb{K}$ *dont le* j*-invariant est égal à* j_0*.*

Preuve : Voir Silverman [Sil09]

□

Définition B.3.54 (points d'une courbe elliptique)**.**

Pour une courbe elliptique E définie sur $\mathbb{K}$, l'ensemble des points $\mathbb{K}$-rationnels de E est :

$$E = \{(x, y) \in \mathbb{K}^2 : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}$$

Soit E une courbe elliptique définie sur $\mathbb{K}$. L'équation de Weierstrass, le discriminant et l'invariant j de E peuvent être simplifiés selon la caractéristique de $\bar{\mathbb{K}}$ (dénommée $\text{Char}(\bar{\mathbb{K}})$). Les versions simplifiées sont données dans le tableau suivant.

Table B.1 – Différentes versions simplifiées de l'équation de Weierstrass

$\text{Char}(\bar{\mathbb{K}})$	Équation simplifiée	Discriminant	j-invariant
$\text{Char}(\bar{\mathbb{K}}) \neq 2, 3$	$E : y^2 = x^3 + ax + b$	$\Delta = -16(4a^3 + 27b^2)$	$j(E) = -1728\frac{(4a)^3}{\Delta}$
$\text{Char}(\bar{\mathbb{K}}) = 2, j(E) = 0$	$y^2 + xy = x^3 + ax + b$	$\Delta = a^3$	$j(E) = 0$
$\text{Char}(\bar{\mathbb{K}}) = 2, j(E) \neq 0$	$y^2 + xy = x^3 + ax^2 + b$	$\Delta = b$	$j(E) = 1/b$
$\text{Char}(\bar{\mathbb{K}}) = 3, j(E) = 0$	$y^2 = x^3 + ax + b$	$\Delta = -a^3$	$j(E) = 0$
$\text{Char}(\bar{\mathbb{K}}) = 3, j(E) \neq 0$	$y^2 = x^3 + ax^2 + b$	$\Delta = -a^3b$	$j(E) = -a^3/b$

Théorème B.3.55

Soit E *une courbe elliptique et soit* $D \in \mathbb{D}$ *un diviseur. Alors il y a un point unique* $P \in E$ *tel que*

$$D \sim P + (\deg(D) - 1)\mathcal{O}$$

Preuve : Voir Enge [Eng01].

□

Théorème B.3.56
Soit E une courbe elliptique. Pour $D \in \mathrm{Pic}^0(E)$, il y a un unique point $P \in E$ tel que

$$D = P - \mathcal{O}$$

L'application $E \to \mathrm{Pic}^0(E) : P \mapsto D = P - \mathcal{O}$ est une bijection.

Preuve : Il découle du théorème précédent.

□

B.3.2 Loi de groupe sur les courbes elliptiques

L'ensemble des points d'une courbe elliptique E a une structure naturelle d'un groupe abélien. Considérons une courbe elliptique E donnée par une équation de Weierstrass. Puisque $E \subset \mathbb{P}^2$ est définie par une équation de degré trois et est constitué des points $P = (x, y)$ satisfaisant l'équation de Weierstrass, ainsi que le point à l'infini $\mathcal{O} = [0, 1, 0]$, toute droite $L \subset \mathbb{P}^2$ coupe E en exactement trois points P,Q,R (pas nécessairement distincts).

Définition B.3.57 (droite tangent)**.**

Soit $P = (X, Y)$ un point sur une courbe elliptique E. Alors, la droite tangente à E au point P est définie comme suit :

$$\frac{\partial E}{\partial x}(P)(x - X) + \frac{\partial E}{\partial y}(P)(y - Y) = 0$$

La loi de composition $\oplus$ sur E est définie par ce qui suit :

•**Loi de composition géométrique** : Soit $P, Q \in E$. Soient L la droite qui passe par P et Q (si $P = Q$, soit L la tangente à E en P), et R le troisième point d'intersection de L avec E. Soit L' la droite passant par R et $\mathcal{O}$. Alors L' intersecte E en R, $\mathcal{O}$, et un troisième point. Nous désignons ce troisième point par $P \oplus Q$.

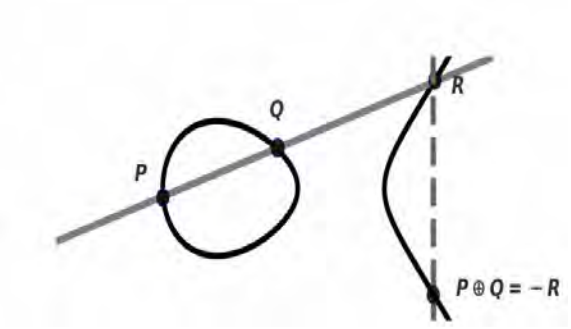
Proposition B.3.58
La loi de composition $\oplus$ sur E a les propriétés suivantes :

1. *$P \oplus \mathcal{O} = P$ pour tout $P \in E$.*
2. *$P \oplus Q = Q \oplus P$ pour tous $P, Q \in E$.*
3. *Soit $P \in E$. Il existe un point de E, noté $-P$, vérifiant $P \oplus (-P) = \mathcal{O}$.*
4. *$(P \oplus Q) \oplus R = P \oplus (Q \oplus R)$ pour tous $P, Q, R \in E$.*

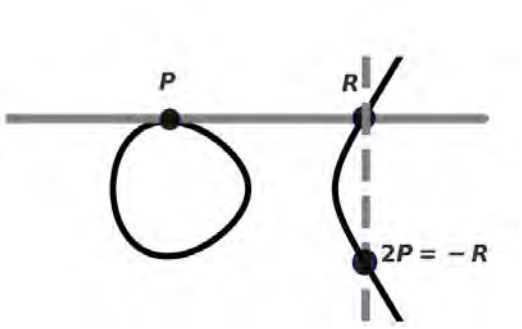
En d'autres termes, $(E, \oplus)$ est un groupe abélien avec comme élément neutre $\mathcal{O}$.

FIGURE B.2 – La loi de composition (réalisé avec SimulaMath [2])

(a) Addition de deux points distincts



(b) Doublement d'un point



Définition B.3.59 (point opposé)**.**

Soit E une courbe elliptique donnée par l'équation de Weierstrass

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

Soit $P = (x, y) \in E$. Le point $(x, -y - a_1x - a_3)$ est appelé le point opposé de E et est noté $-P$. Pour le point à l'infini, on a $-\mathcal{O} = \mathcal{O}$.

•**Loi algébrique de groupes pour l'équation longue de Weierstrass** : Considérons une courbe elliptique donnée par l'équation longue de Weierstrass

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

On décrit la loi d'addition $\oplus$ sur E dans les trois algorithmes suivants, à savoir

— **AdditionDistinct** : pour le calcul de $P \oplus Q$ où $P, Q \in E$, $P \neq \pm Q$ et $P, Q \neq \mathcal{O}$.
— **Doublement** : pour calculer $P \oplus P = 2P$ avec $P \in E$, $P \neq Q$.
— **addition** : pour le calcul de $P \oplus Q$ où P, Q sont des points arbitraires sur E.

Algorithme 19 : AdditionDistinct(P, Q)

Entrées : $E, P = (x_1, y_1)$ et $Q = (x_2, y_2)$ avec $P \neq \pm Q$
Sortie : $P \oplus Q = (x_3, y_3)$

1 $\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$;
2 $x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2$;
3 $y_3 = \lambda(x_1 - x_3) - y_1 - (a_1x_3 + a_3)$;
4 **Retourner** $P \oplus Q = (x_3, y_3)$

L'algorithme de doublement est le suivant.

2. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

Algorithme 20 : Doublement(P)

Entrées : E et $P = (x_1, y_1) \in E$
Sortie : $2P = P \oplus P = (x_2, y_2)$

$\lambda = \dfrac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}$;
$x_2 = \lambda^2 + a_1\lambda - a_2 - 2x_1$;
$y_2 = \lambda(x_1 - x_2) - y_1 - (a_1x_2 + a_3)$;
Retourner $2P = (x_2, y_2)$

L'algorithme d'addition pour tout point P et Q est le suivant.

Algorithme 21 : Addition(P, Q) (P et Q ne sont pas nécessairement distincts)

Entrées : $E, P \in E$ and $Q \in E$
Sortie : $P \oplus Q = R$

$R = \mathcal{O}$;
Si $P = \mathcal{O}$ **Alors**
  $R = Q$;
Sinon Si $Q = \mathcal{O}$ **Alors**
  $R = P$;
Sinon Si $P = -Q$ **Alors**
  $R = \mathcal{O}$;
Sinon Si $P \neq \pm Q$ **Alors**
  $R =$**AdditionDistinct**(P, Q) ;
Sinon $//P = Q$
  $R =$**Doublement**(P) ;
Fin Si
Retourner $P \oplus Q = R$

•**Loi algébrique de groupes pour l'équation courte de Weierstrass** : Considérons une courbe elliptique donnée par l'équation courte de Weierstrass $E : y^2 = x^3 + ax + b$.

Nous décrivons la loi d'addition (simplifiée) $\oplus$ sur E dans les trois algorithmes suivants.

Algorithme 22 : AdditionDistinct(P, Q)

Entrées : $E : y^2 = x^3 + ax + b, P = (x_1, y_1)$ et $Q = (x_2, y_2)$ avec $P \neq \pm Q$
Sortie : $P \oplus Q = (x_3, y_3)$

$\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$;
$x_3 = \lambda^2 - x_1 - x_2$;
$y_3 = \lambda(x_1 - x_3) - y_1$;
Retourner $P \oplus Q = (x_3, y_3)$

Algorithme 23 : Doublement(P)

Entrées : $E : y^2 = x^3 + ax + b$ et $P = (x_1, y_1) \in E$
Sortie : $2P = P \oplus P = (x_2, y_2)$

$\lambda = \dfrac{3x_1^2 + a}{2y_1}$;
$x_2 = \lambda^2 - 2x_1$;
$y_2 = \lambda(x_1 - x_2) - y_1$;
Retourner $2P = (x_2, y_2)$

•**Multiplication d'un point par un scalaire :** Dans les deux algorithmes suivants, nous décrivons comment calculer efficacement $kP = \underbrace{P \oplus P + \oplus \ldots \oplus P}_{\text{k fois}}, k \geqslant 0$ en utilisant la méthode m-ary ou la méthode binaire.

Algorithme 24 : Multiplication d'un point par un scalaire : Méthode binaire.

Entrées : $P \in E$ and an l-bit integer $k = \sum_{j=0}^{l-1} k_j 2^j, k_j \in \{0, 1\}$ avec $k_{l-1} = 1$
Sortie : $R = kP$

$R = P$;
Pour $j = l - 1$ *à* 0 *par pas de* -1 **Faire**
  $R =$**Doublement**(R) ; /* $R := 2R$ */
  Si $k_j = 1$ **Alors**
    $R =$**Addition**(R, P); /* $R := R \oplus P$ */
  Fin Si
Fin Pour
Retourner $R = kP$

Notons que si k est un entier négatif, $kP = (-k)(-P)$

Algorithme 25 : Multiplication d'un point par un scalaire : Méthode m-ary.

Entrées : $P \in E$, $m = 2^r (r \in \mathbb{N}^*)$, et un entier de l-bits
$k = \sum_{j=0}^{l-1} k_j m^j, k_j \in \{0, 1, \ldots, m-1\}$ avec $k_{l-1} \neq 0$
Sortie : $R = kP$

```
/* Calculs préalables                                          */
```
1 $P_0 = \mathcal{O}$;
2 $P_1 = P$;
3 **Pour** $i = 2$ *à* m **Faire**
4 | P_i =**Addition**(P_{i-1}, P); /* $P_i = P_{i-1} \oplus P$ */
5 **Fin Pour**
```
/* On a P_i = iP                                               */
/* Boucle principale                                           */
```
6 $R = \mathcal{O}$;
7 **Pour** $j = l - 1$ *à* 0 *par pas de* -1 **Faire**
8 | $R = mR$; /* (Cela nécessite r doublements) */
9 | R =**Addition**(R, P_{k_j}); /* $R = R \oplus P_{k_j}$ */
10 **Fin Pour**
11 **Retourner** kP

La méthode binaire est un cas particulier de la méthode m-ary (elle correspond à $r = 1$) .

Théorème B.3.60 (Mordell-Weil)
Soit E une courbe elliptique définie sur un corps de nombre $\mathbb{K}$. Le groupe $E(\mathbb{K})$ est finement généré. Autrement dit, il existe des points $P_1, P_2, \ldots, P_k \in E(\mathbb{K})$ tels que chaque élément de $E(\mathbb{K})$ est de la forme $n_1P_1 \oplus n_2P_2 \oplus \ldots \oplus n_kP_k$ pour certains entiers $n_1, n_2, \ldots, n_k \in \mathbb{Z}$.

B.3.3 Courbes elliptiques sur corps fini

Dans cette section, nous supposons que $\mathbb{K}$ est un corps fini contenant q éléments où q est une puissance d'un nombre premier p, i.e $\mathbb{K} = \mathbb{F}_q$.

Définition B.3.61 (ordre)**.**

Soit E une courbe elliptique définie sur $\mathbb{F}_q$.
- Le nombre de points dans $E(\mathbb{F}_q)$, noté $\#E$, est appelé l'ordre de E sur $\mathbb{F}_q$.
- Soit P un point de E. L'ordre de P est le plus petit entier $k \geqslant 1$ tel que $kP = \mathcal{O}$. C'est l'ordre du sous-groupe de E engendré par P.

Remarque B.3.62
Un point de E est un point d'ordre 2 si et seulement si $P = -P$. De plus, si E est défini par l'équation courte de Weierstrass $E : y^2 = x^3 + ax + b$ sur $\mathbb{F}_q$, un point d'ordre 2 est sous la forme $(x, 0)$.

Théorème B.3.63 (Hasse)
Soit E une courbe elliptique définie sur $\mathbb{F}_q$. Alors

$$q + 1 - 2\sqrt{q} \leqslant \#E \leqslant q + 1 + 2\sqrt{q}$$

*L'intervalle $[q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}]$ est appelé intervalle de **Hasse**.*

Preuve : Voir [Was03].

□

Définition B.3.64 (twist, twist quadratique)**.**

Soit E une courbe elliptique sur $\mathbb{F}_q$. Une courbe elliptique E' sur $\mathbb{F}_q$ est un **twist de degré** d de E s'il existe un isomorphisme $\phi_d : E \to E'$ défini sur $\mathbb{F}_{q^d}$ et tel que d soit minimal. Si $d = 2$, on dit que E' est un **twist quadratique** de E.

Exemple B.3.3 *Chaque courbe elliptique $E : y^2 = x^3 + ax + b$ sur $\mathbb{F}_q$ a un twist quadratique $E' : \delta y^2 = x^3 + ax + b$ pour tout $\delta \in \mathbb{F}_q$ qui n'est pas carré.*

Définition B.3.65 (endomorphisme de Frobenius)**.**

Soit E une courbe elliptique définie sur $\mathbb{F}_q$. L'endomorphisme $\pi : E \to E : (x, y) \mapsto (x^q, y^q)$ et $\pi(\mathcal{O}) = \mathcal{O}$ est appelé endomorphisme de Frobenius.

Proposition B.3.66
Soit E une courbe elliptique définie sur $\mathbb{F}_q$, et soit $(x, y) \in E(\bar{\mathbb{F}}_q)$.

1. *$\pi(x, y) \in E(\bar{\mathbb{F}}_q)$*
2. *$(x, y) \in E(\mathbb{F}_q)$ si et seulement si $\pi(x, y) = (x, y)$*

Preuve : Voir [Was03].

□

Proposition B.3.67
Soit E une courbe elliptique définie sur $\mathbb{F}_q$. Alors π est un endomorphisme de E de degré q, et π n'est pas séparable.

Le Frobenius satisfait une équation quadratique $\pi^2 + q = t\pi$ où t est la trace de π.

Théorème B.3.68
Soit E une courbe elliptique sur $\mathbb{F}_q$. Alors $\#E(\mathbb{F}_q) = q + 1 - t$

Preuve : Voir [Was03].

□

Définition B.3.69 (courbe elliptique supersingulière et ordinaire)**.**

Une courbe elliptique définie sur $\mathbb{F}_q, q = p^n$ est supersingulière si p divise la trace $t = \#E(\mathbb{F}_q) - (q+1)$. Une courbe elliptique non supersingulière est appelée une courbe elliptique ordinaire.

Proposition B.3.70

1. *Si $p \in \{2,3\}$, alors E est supersingulière si et seulement si $j(E) = 0$.*
2. *Supposons que $p \geqslant 5$ est un nombre premier et que $E : y^2 = x^3 + ax + b$ est défini sur $\mathbb{F}_p$. Alors E est supersingulière si et seulement si $a = 0$, ce qui est le cas si et seulement si $\#E(\mathbb{F}_p) = p + 1$.*
3. *Supposons que q est impair et $q \equiv 2 \bmod 3$. Soit $b \in \mathbb{F}_q^*$. Alors la courbe elliptique E donnée par $y^2 = x^3 + b$ est supersingulière.*

Preuve : Voir [Was03].

□

Théorème B.3.71
Soit E une courbe elliptique sur $\mathbb{F}_q$. Alors $\#E(\mathbb{F}_{q^n}) = q^n + 1 - V_n$ pour tout $n \geqslant 2$, où $\{V_n\}$ est la séquence définie récursivement par $V_0 = 0, V_1 = t$ et $V_k = V_1 V_{k-1} - qV_{n-2}$ où t est la trace de Frobenuis.

Preuve : Voir [Was03].

□

Cette propriété est très utile dans le comptage de points d'une courbe elliptique définie sur un corps fini $\mathbb{F}_{p^n}$ non premier i.e $n > 1$.

B.3.4 Différentes formes de courbes elliptiques

Définition B.3.72 (courbes birationnellement équivalentes)**.**

Deux courbes algébriques sont dites **birationnellement équivalentes** si leurs corps de fonctions sont isomorphes. On appelle **un modèle**, la représentation d'une classe birationnelle.

Modèle de Montgomery

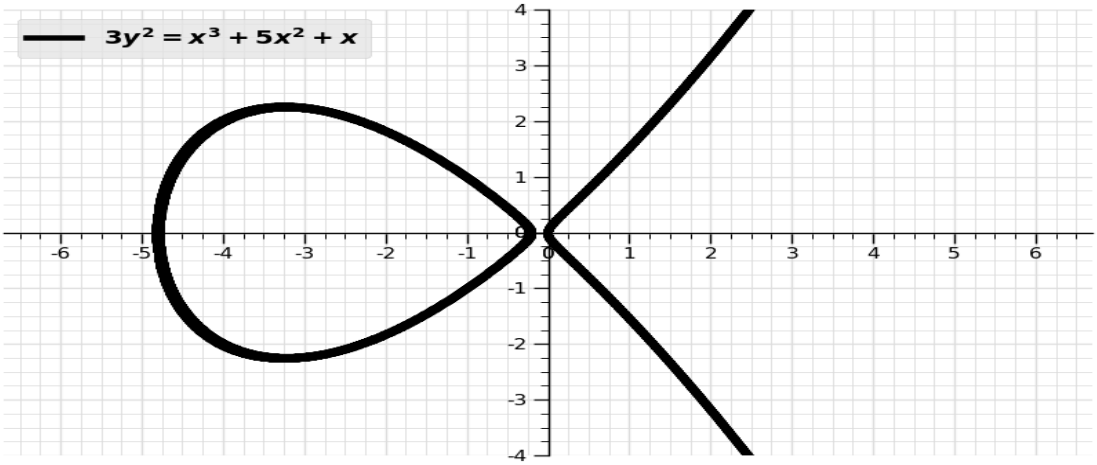
Définition B.3.73 (courbe de Montgomery)**.**

Soient $A, B \in \mathbb{K}$ des éléments satisfaisant $B(A^2-4) \neq 0$ dans $\mathbb{K}$ (où $\text{char}(\mathbb{K}) \neq 2$). Une courbe de Montgomery $E_{A,B}$ [Mon87] définie sur $\mathbb{K}$, notée $E_{A,B}/\mathbb{K}$, est définie comme étant l'ensemble des points $P = (x, y) \in \mathbb{K} \times \mathbb{K}$ qui sont solutions de l'équation

$$By^2 = x^3 + Ax^2 + x$$

et un point supplémentaire, appelé le point à l'infini.

FIGURE B.3 – courbe de Montgomery : $3y^2 = x^3+5x^2+x$ sur $\mathbb{R}$ (réalisé avec SimulaMath [3])



Définition B.3.74 (j-invariant d'une courbe de Montgomery)**.**

Le j-invariant d'une courbe de Montgomery $E_{A,B}$ est défini par

$$j(E_{A,B}) = \frac{256(A^2-3)^3}{A^2-4}$$

Remarque B.3.75

- $P = (0, 0)$ *est toujours un point de* $E_{A,B}$ *et est d'ordre* 2.
- *Il faut noter que la classe des* $\mathbb{F}_q$*-isomorphismes de* $E_{A,B}$ *est entièrement déterminée par* A^2*, et est indépendante de* B*. Cela implique que toutes les courbes elliptiques sur* $\mathbb{F}_q$ *n'ont pas un modèle de Montgomery sur* $\mathbb{F}_q$ *puisque chaque élément de* $\mathbb{F}_q$ *est le* j*-invariant d'une courbe sur* $\mathbb{F}_q$*.*

3. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

Les opérations sur la courbe elliptique sous forme de Montgomery : $E_{A,B} : By^2 = x^3 + Ax^2 + x$ pour les coordonnées affines sont les suivantes.

Soient $P = (x_1, y_1), Q = (x_2, y_2)$ des points de $E_{A,B}$. Alors, le point $R = (x_3, y_3) = P + Q$ est donné par les algorithmes suivants :

— On suppose que $P \neq \pm Q$.

Algorithme 26 : Addition pour la forme de Montgomery

Entrées : $E_{A,B} : By^2 = x^3 + Ax^2 + x, P = (x_1, y_1)$ et $Q = (x_2, y_2)$ avec $P \neq \pm Q$
Sortie : $P \oplus Q = (x_3, y_3)$

1 $\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$;
2 $x_3 = B\lambda^2 - A - x_1 - x_2$;
3 $y_3 = \lambda(x_1 - x_3) - y_1$;
4 **Retourner** $P \oplus Q = (x_3, y_3)$

— On suppose maintenant que $P = Q$ et $y_1 \neq 0$.

Algorithme 27 : Doublement sur une courbe de Montgomery)

Entrées : $E_{A,B} : By^2 = x^3 + Ax^2 + x$ and $P = (x_1, y_1) \in E$)
Sortie : $2P = P \oplus P = (x_3, y_3)$

1 $\lambda = \dfrac{3x_1^2 + 2Ax_1 + 1}{2By_1}$;
2 $x_3 = B\lambda^2 - A - 2x_1$;
3 $y_3 = \lambda(x_1 - x_3) - y_1$;
4 **Retourner** $2P = (x_3, y_3)$

Notons que le point opposé de $P = (x, y) \in E_{A,B}$ est $-P = (x, -y)$ et donc $P - P = \mathcal{O}$. Montgomery remarque dans [Mon87] que l'ordre de $E_{A,B}(\mathbb{F}_q)$ est toujours divisible par 4.

Proposition B.3.76
Soit $E_{A,B} : By^2 = x^3 + Ax^2 + x$ une courbe de Montgomery définie sur $\mathbb{F}_q$. Nous avons ce qui suit :

1. *Si $B(A+2)$ est un carré dans $\mathbb{F}_q$ alors $(1, \pm\sqrt{(A+2)/B})$ sont des points d'ordre 4 dans $E_{A,B}(\mathbb{F}_q)$.*
2. *De façon similaire, si $B(A-2)$ est un carré dans $\mathbb{F}_q$ alors $(1, \pm\sqrt{(A-2)/B})$ sont des points d'ordre 4 dans $E_{A,B}(\mathbb{F}_q)$.*
3. *Si ni $B(A+2)$ ni $B(A-2)$ n'est un carré, alors $A^2 - 4$ doit être un carré, alors $x^2 + Ax + 1$ se factorise complètement sur $\mathbb{F}_q$, donc $E_{A,B}$ a un point de torsion rationnel d'ordre 2.*

Preuve : Voir [OKS00].

□

Modèle d'Edwards

— En 2007, Edwards [Edw07] a introduit une nouvelle forme de courbes elliptiques sur un corps non binaire $\mathbb{K}$: courbes données par une équation de la forme

$$E_c : x^2 + y^2 = c^2(1 + x^2y^2) \tag{B.2}$$

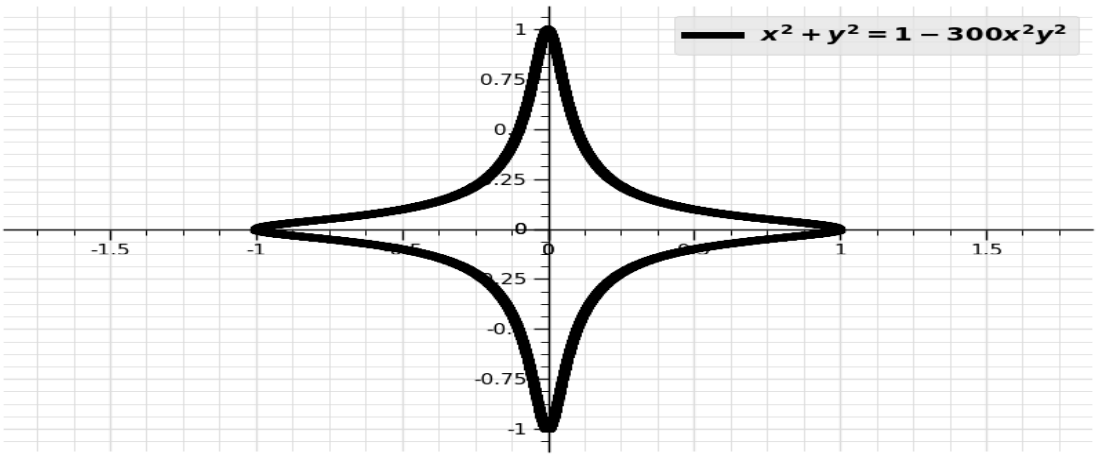
Il a également montré que chaque courbe elliptique sur un corps non binaire $\mathbb{K}$ peut être transformée en une courbe d'Edwards si $\mathbb{K}$ est algébriquement clos. Cependant, sur un corps fini $\mathbb{F}_q$, seul un petit nombre de courbes elliptiques peuvent être exprimées sous cette forme.

— Bernstein et Lange, dans [BL07], ont généralisé l'équation des courbes d'Edwards Eq. B.2 à l'équation

$$x^2 + y^2 = c^2(1 + dx^2y^2) \tag{B.3}$$

où $cd(1 - dc^4) \neq 0$ sont dans $\mathbb{K}$ qui couvrent plus de courbes elliptiques. Il faut noter que tous les courbes elliptiques définies par Eq. B.3 sont isomorphes aux courbes $x^2 + y^2 = 1 + dx^2y^2$.

FIGURE B.4 – courbe d'Edwards : $x^2+y^2 = 1-300x^2y^2$ sur $\mathbb{R}$ (réalisé avec SimulaMath [4])



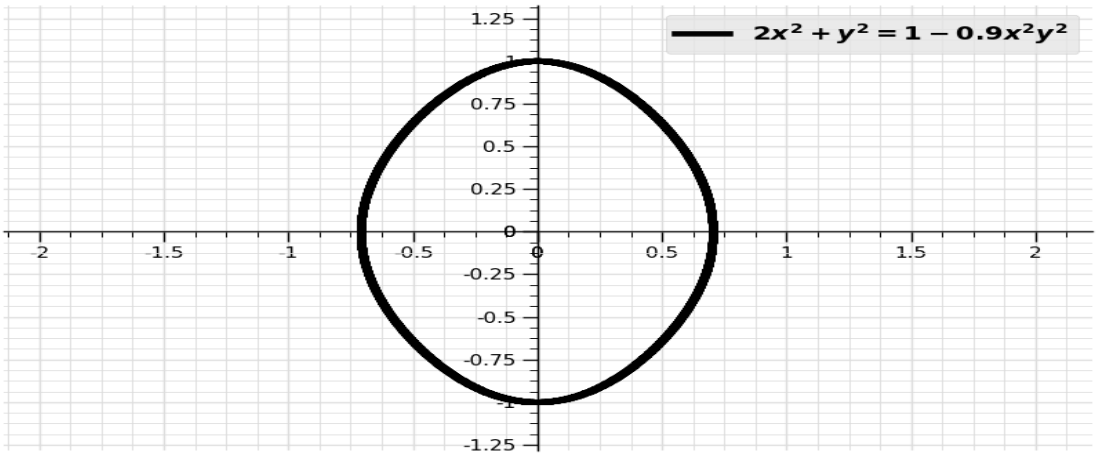
— En 2008, Bernstein et al. [BBJ+08] ont introduit les courbes d'Edwards tordues

$$E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2 \tag{B.4}$$

dans un corps non binaire $\mathbb{K}$ qui généralisent les courbes elliptiques précédentes.

4. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

FIGURE B.5 – courbe d'Edwards : $2x^2+y^2 = 1-0.9x^2y^2$ sur $\mathbb{R}$ (réalisé avec SimulaMath [5])



Remarque B.3.77
Par une courbe d'Edwards, nous voulons dire que c'est une courbe de la forme $x^2 + y^2 = 1 + dx^2y^2$ *et par une twist d'Edwards (ou courbe d'Edwards tordue), c'est une courbe donnée par* $E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2$. *On pourrait considérer l'équation plus générale* $E_{a,d,c} : ax^2 + y^2 = c^2(1 + dx^2y^2)$ *mais elle est toujours isomorphe à une courbe d'Edwards tordue.*

Nous présentons l'addition pour les courbes d'Edwards tordues $E_{a,d} : ax^2+y^2 = 1+dx^2y^2$ définies sur $\mathbb{K}$ avec $\text{char}(\mathbb{K}) \neq 2$. En fait une courbe d'Edwards est un cas particulier, il suffit de prendre $a = 1$.

— L'élément neutre est $(0, 1)$, et l'opposé de (x_1, y_1) est $(-x_1, y_1)$.

Algorithme 28 : Ajout sur une courbe d'Edwards tordue $E_{a,d}$

Entrées : $E_{a,d} : ax^2 + y^2 = 1 + dx^2y^2$, $P = (x_1, y_1)$ et $Q = (x_2, y_2)$
Sortie : $P \oplus Q = (x_3, y_3)$

— 1 $x_3 = \dfrac{x_1y_2 + x_2y_1}{1 + dx_1x_2y_1y_2}$;
2 $y_3 = \dfrac{y_1y_2 - ax_1x_2}{1 - dx_1x_2y_1y_2}$;
3 **Retourner** $P \oplus Q = (x_3, y_3)$

Nous avons les résultats suivants.

Théorème B.3.78
Soit $\mathbb{K}$ *un corps fini avec* $\text{char}(\mathbb{K}) \neq 2$. *Soit* E *une courbe elliptique sur* $\mathbb{K}$. *Le groupe* $E(\mathbb{K})$ *a un élément d'ordre* 4 *si et seulement si* E *est birationnellement équivalente sur* $\mathbb{K}$ *à une courbe d'Edwards.*

Preuve : Voir [BBJ+08].

□

5. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

Théorème B.3.79
Soit $\mathbb{K}$ *un corps avec* $\mathrm{char}(\mathbb{K}) \neq 2$. *Toute courbe elliptique sur* $\mathbb{K}$ *ayant trois points* $\mathbb{K}$*-rationnels d'ordre* 2 *est* 2*-isogène*[6] *sur* $\mathbb{K}$ *à une twist d'Edwards.*

Preuve : Voir [BBJ+08].

□

Théorème B.3.80
Soit $\mathbb{K}$ *un corps de caractéristique* $\mathrm{char}(\mathbb{K}) \neq 2$.

- *Toute twist d'Edwards sur* $\mathbb{K}$ *est birationnellement équivalente sur* $\mathbb{K}$ *à une courbe de Montgomery.*
- *Inversement, toute courbe de Montgomery sur* $\mathbb{K}$ *est birationnellement équivalente sur* $\mathbb{K}$ *à une twist d'Edwards.*

Preuve : Voir [BBJ+08].

□

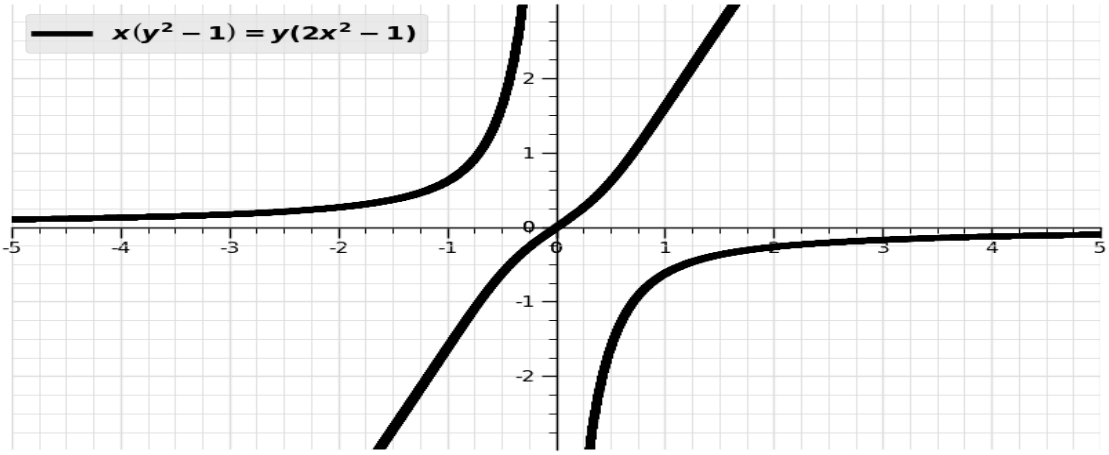
Modèle de Huff

Les courbes de Huff, introduites pour la première fois par Huff [Huf48] en 1948, ont été utilisées par Joye, Tibouchi et Vergnaud [JTV10] pour développer un modèle de courbe elliptique :

$$ax(y^2 - 1) = by(x^2 - 1)$$

sur un corps fini $\mathbb{K}$ où $\mathrm{char}(\mathbb{K}) > 2$. Ils ont également présenté les formules explicites efficaces pour additionner ou doubler des points sur les courbes de Huff.

FIGURE B.6 – Courbe de Huff : $x(y^2 - 1) = y(2x^2 - 1)$ sur $\mathbb{R}$ (réalisé avec SimulaMath[7])



6. Les isogénies sont définies à la sous-section B.3.5
7. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

Ciss et Sow [CS11], en 2011, ont introduit les courbes de Huff généralisées :

$$ax(y^2 - c) = by(x^2 - d)$$

avec $abcd(a^2c - b^2d) \neq 0$, qui contiennent des courbes de Huff classiques [JTV10] comme cas particuliers. Indépendamment, Wu et Feng [WF12] ont présenté un autre type de courbes qu'ils ont aussi appelé courbes de Huff généralisées :

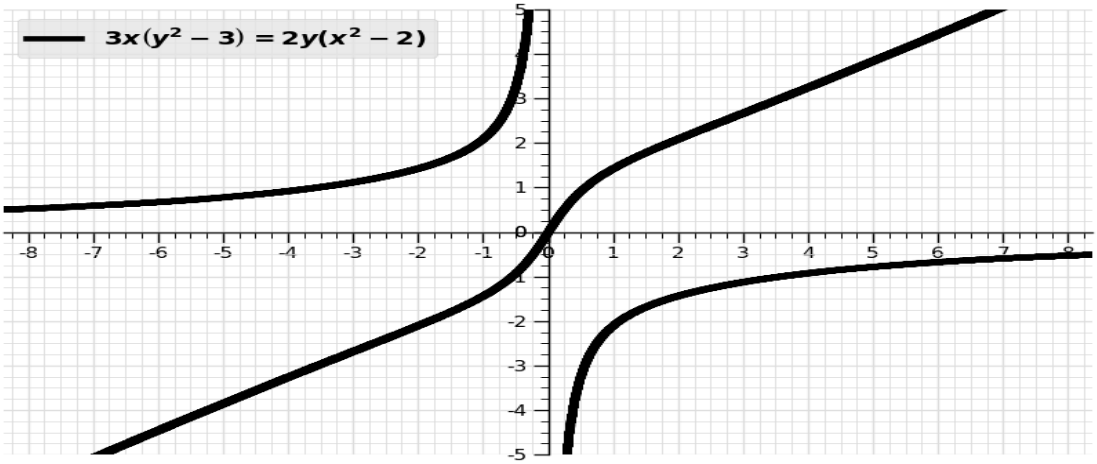
$$x(ay^2 - 1) = y(bx^2 - 1)$$

qui est en fait une variation équivalente de la construction de Ciss et Sow. Toujours la même année, Devigne et Joye [DJ11] ont également analysé des courbes de Huff sur des corps binaires :

$$ax(y^2 + cy + 1) = by(x^2 + cx + 1)$$

avec $abc(a - b) \neq 0$.

FIGURE B.7 – Courbe de Huff : $3x(y^2 - 3) = 3y(x^2 - 2)$ sur $\mathbb{R}$ (réalisé avec SimulaMath [8])



Autres modèles de courbes

Il existe également de nombreux autres modèles de courbes elliptiques à savoir :

— **modèle Hessien** $E : x^3 + y^3 + 1 = 3dxy$ avec $d^3 \neq 1$ introduit par Hesse [Hes44] en 1844.

— la quartique de **Jacobi** $E_b : y^2 = x^4 + 2bx^2 + 1$ sur un corps non binaire $\mathbb{K}$ étendu ultérieurement sous la forme $E_{d,b} : dy^2 = x^4 + 2bx^2 + 1$ [HWCD09].

— L'intersection tordue de **Jacobi** $au^2 + v^2 = 1, bv^2 + w^2 = 1$ avec $ab(a - b) \neq 0$ introduit par [FNW10]. La courbe elliptique d'intersection de Jacobi est une courbe d'intersection de Jacobi tordue avec $a = 1$.

8. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

B.3.5 Isogénies

Définition B.3.81 (isogénie)**.**

Soient E_1 et E_2 deux courbes elliptiques. Une isogénie [Feo17] de E_1 à E_2 est un morphisme non constant

$$\phi : E_1 \to E_2$$

où $\phi(\mathcal{O}) = \mathcal{O}$. Si une telle application existe, on dira que E_1 est isogène à E_2.

Remarque B.3.82

1. *Une isogénie est aussi un homomorphisme de groupe, c'est-à-dire* $\phi(P + Q) = \phi(P) + \phi(Q)$ *pour tous* $P, Q \in E_1$.
2. *Une isogénie non constante est surjective, c'est-à-dire* $\phi(E_1) = E_2$.
3. *Pour les applications en cryptographie, on s'intéresse aux isogénies non constantes.*

Exemple B.3.4

1. *L'application qui envoie chaque point de* E_1 *à* $\mathcal{O} \in E_2$ *est aussi appelée une isogénie. C'est l'isogénie nulle, et c'est la seule isogénie constante.*
2. *L'application* $[n] : E \to E : P \mapsto nP$ *est une isogénie.*

Si $\phi : E_1 \to E_2$ est une isogénie non constante alors ϕ induit une injection de corps de fonctions qui laisse invariant $\bar{\mathbb{K}}$, noté ϕ^*, défini de $\bar{\mathbb{K}}(E_2)$ à $\bar{\mathbb{K}}(E_1)$ par

$$\phi^*(f) = f \circ \phi$$

Définition B.3.83 (degré d'une isogénie)**.**

Le degré d'une isogénie $\phi : E_1 \to E_2$, noté $\deg(\phi)$, est défini comme le degré de l'extension finie $\bar{\mathbb{K}}(E_1)/\phi^*(\bar{\mathbb{K}}(E_2))$, où $\mathbb{K}(*)$ est le corps de fonction de la courbe, et ϕ^* est l'application entre les corps de fonction induits par l'isogénie ϕ. Par convention, on pose $\deg([0]) = 0$. Une isogénie de degré l est appelée une l-isogénie.

Exemple B.3.5

1. *L'endomorphisme de Frobenius* π *sur* $\mathbb{F}_q$ *est de degré* q.

Définition B.3.84 (endomorphisme-isomorphisme)**.**

1. Un endomorphisme est un homomorphisme d'une courbe à elle-même. Les endomorphismes de E forment un anneau noté $\mathrm{End}(E)$.
2. Un isomorphisme est une isogénie de degré un.
3. Si $\mathrm{End}(E)$ contient un autre endomorphisme que $[n]$, alors on dit que E a une multiplication complexe.

Proposition B.3.85
Pour toute chaîne $E_1 \xrightarrow{\phi} E_2 \xrightarrow{\psi} E_3$, *on a* $\deg(\phi \circ \psi) = \deg(\phi)\deg(\psi)$. *En particulier* $\deg([n]) = n^2$ *pour tout* $n > 0$.

Théorème B.3.86
Soit $\phi : E_1 \to E_2$ *une isogénie non constante de degré* n. *Alors, il existe une isogénie unique de degré* $\widehat{\phi} : E_2 \to E_1$ *telle que* $\phi \circ \widehat{\phi} = [n]$ *sur* E_1 *et* $\widehat{\phi} \circ \phi = [n]$ *sur* E_2. *On appelle* $\widehat{\phi}$ *l'isogénie duale à* ϕ. *Nous avons aussi,* $\deg(\phi) = \deg(\widehat{\phi})$.

Preuve : Voir [Sil09].

□

Définition B.3.87 (noyau d'une isogénie)**.**

Soit $\phi : E_1 \to E_2$ une isogénie. Le noyau de ϕ est défini par $\ker(\phi) = \{P \in E_1 : \phi(P) = \mathcal{O}\}$

Proposition B.3.88
Soit $\phi : E_1 \to E_2$ *une isogénie séparable. Alors* $\deg(\phi) = \#\ker(\phi)$

Définition B.3.89 (sous-groupe de n-torsion)**.**

On définit le sous-groupe de n-torsion d'une courbe elliptique E comme $E[n] = \{P \in E : nP = \mathcal{O}\}$

• **Isogénie sur un corps fini** : Si ϕ est une isogénie sur $\mathbb{F}_q$, on peut l'exprimer en termes de deux applications rationnelles f et g sur $\mathbb{F}_q$ telles que $\phi(x, y) = (f(x) = p(x)/q(x), yg(x))$ où les polynômes $p(x)$ et q(x) sur $\mathbb{F}_q$ n'ont pas de facteur commun. Dans ce cas,

$$\deg(\phi) = \max\{\deg(p(x)), \deg(q(x))\}$$

Théorème B.3.90
Pour tout sous-groupe fini G *de* $E(\mathbb{F}_q)$, *il existe une isogénie unique (à l'isomorphisme près)* $\phi : E \to E'$ *tel que* $\ker(\phi) = G$ *et* $deg(\phi) = \#G$. *Dans ce cas,* E' *est noté par* E/G.

Il faut noter qu'on peut calculer E/G en utilisant les formules du Velu [Vel71].

On peut construire une isogénie entre les courbes elliptiques E et E' sans utiliser un sous-groupe de E. Kohel [Koh96] a proposé que cette isogénie peut être calculée à partir du polynôme du noyau.

• Les formules de Velu [Vel71] : Soit E une courbe elliptique donnée par l'équation généralisée de Weierstrass

$$E : 2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

avec a_i un élément d'un corps fini $\mathbb{K}$. Soit G un sous-groupe fini de $E(\bar{\mathbb{K}})$. On sait qu'il existe une courbe elliptique E_2 et une isogénie séparable ϕ de E à E_2 telle que $G = \ker(\phi)$.

Nous allons expliciter l'équation de E_2 (voir [Was03] pour plus de détails). Pour un point $Q = (x_Q, y_Q) \in E$ avec $Q \neq \mathcal{O}$, on définit

$$\begin{aligned} g_Q^x &= 3x_Q^2 + 2a_2x_Q + a_4 - a_1y_Q \\ g_Q^y &= -2y_Q - a_1x_Q - a_3 \\ v_q &= \begin{cases} g_Q^x & \text{if ord}(Q) = 2 \\ 2g_Q^x - a_1g_Q^y & \text{Sinon} \end{cases} \\ u_Q &= (g_Q^y)^2 \end{aligned}$$

Soit G_2 l'ensemble des points d'ordre 2 dans G. Soit $R \in G$ tel que l'on ait une union disjointe

$$G = \{\mathcal{O}\} \cup G_2 \cup R \cup (-R)$$

en d'autres termes, pour chaque paire de points de 2-torsion $P, -P \in G$, on met exactement un d'eux dans R). Soit $S = R \cup G_2$. Soit $v = \sum_{Q \in S} v_Q, \sum_{Q \in S}(u_Q + x_Q v_Q)$. Alors E_2 a comme équation

$$Y^2 + A_1XY + A_3Y = X^3 + A_2X^2 + A_4X + A_6$$

où $A_1 = a_1, A_2 = a_2, A_3 = a_3, A_4 = a_4 - 5v, A_6 = a_6 - (a_1^2 + 4a_2)v - 7w$
L'isogénie $\phi : E_1 \to E_2 : (x, y) \to (X, Y)$ est donnée par

$$X = x + \sum_{Q \in S}\left(\frac{v_Q}{x - x_Q} + \frac{u_Q}{(x - x_Q)^2}\right)$$

$$Y = y - \sum_{Q \in S}\left(u_Q\frac{2y + a_1x + a_3}{(x - x_Q)^3} + v_Q\frac{a_1(x - x_Q) + y - y_Q}{(x - x_Q)^2} + \frac{a_1u_Q - g_Q^xg_Q^y}{(x - x_Q)^2}\right)$$

B.3.6 ECDH et ECDSA

ECDH : Echange de clefs Diffie-Helman sur les courbes elliptiques

Alice et Bob veulent se mettre d'accord sur une clef commune qu'ils peuvent utiliser pour échanger des données via un schéma de chiffrement symétrique tel que AES. Ils peuvent procéder comme suit :

1. Alice et Bob s'accordent sur une courbe elliptique E sur un corps fini $\mathbb{F}_q$ tel que le problème du logarithme discret est difficile à résoudre dans $E(\mathbb{F}_q)$. Ils sont aussi tombés d'accord sur un point $P \in E(\mathbb{F}_q)$ tel que le sous-groupe généré par P ait un ordre grand (généralement, la courbe et le point sont choisis de telle sorte que l'ordre soit un grand nombre premier).
2. Alice choisit un entier secret **a**, calcule $P_a = aP$, et envoie P_a à Bob.
3. Bob choisit un entier secret **b**, calcule $P_b = bP$, et envoie P_b à Alice.
4. Alice calcule $aP_b = abP$.

5. Bob calcule $bP_a = baP$
6. Alice et Bob utilisent une méthode convenue publiquement pour extraire une clef de abP. Par exemple, ils pourraient utiliser les 256 derniers bits de la coordonnée x de abP comme clef. Ils peuvent aussi évaluer une fonction de hachage en utilisant la coordonnée x..

Les seules informations que l'espion Ève voit sont la courbe E, le corps fini $\mathbb{F}_q$, et les points P, aP et bP. Elle doit donc résoudre le problème suivant [Was03] :

1. **problème calculatoire de Diffie-Hellman** :
   Étant donnés P, aP, et bP dans $E(\mathbb{F}_q)$, déterminer abP.
2. **Problème décisionnel de Diffie-Hellman** :
   Étant donnés P, aP, et bP dans $E(\mathbb{F}_q)$, et étant donné un point $Q \in E(\mathbb{F}_q)$, déterminer si oui ou non $Q = abP$.

Remarque B.3.91
Le problème calculatoire de Diffie-Hellman et le problème décisionnel de Diffie-Hellman peuvent être sur des groupes arbitraires. À l'origine, ils sont apparus dans le contexte de groupes multiplicatifs $\mathbb{F}_q^*$ de corps finis.

ECDSA : Elliptic Curve Digital Signature Algorithm

L'algorithme de signature ECDSA (Elliptic Curve Digital Signature Algorithm) est basé sur l'algorithme de signature numérique (DSA). La version originale du DSA utilisait des groupes multiplicatifs de corps finis alors que la version ECDSA utilise des courbes elliptiques.

•**Algorithme de génération de clefs** : Soit p un grand nombre premier et soit E une courbe elliptique définie sur $\mathbb{F}_p$ telle que $\#E(\mathbb{F}_p) = fq$, où q est un grand nombre premier et f est un petit entier, généralement $1, 2$, ou 4 (f doit être petit pour que l'algorithme reste efficace).
Si Alice veut signer un document m,

— Elle choisit un point A sur E ayant un ordre premier q, tel que le problème du Logarithme Discret dans $< A >$ soit infaisable.
— Enfin, Alice choisit un entier secret x et calcule $B = xA$.

Alors Alice rend publique l'information suivante :

$$\mathcal{K} = \{p, q, E, A, B\}$$

La clef privée d'Alice est $\mathbf{sk} = \mathbf{x}$, elle la garde secrète.

•**Algorithme de signature** : Pour signer le message m, Alice utilise l'algorithme suivant :

Algorithme 29 : Algorithme de signature

Entrées : Données publiques $\mathcal{K} = \{p, q, E, A, B\}$, le message m et la clef privée $sk = x$)

Sortie : (r, s) : la signature du message m

1 Choisir un entier aléatoire k avec $1 \leqslant k \leqslant q - 1$;
2 Calculer $kA = (u, v)$;
3 Calculer $r = u(\bmod q)$;
4 Calculer $s = k^{-1}(H(m) + xr)(\bmod q)$; `/* H est une fonction de hachage cryptographique */`
5 **Si** $r = 0$ *ou* $s = 0$ **Alors**
6 | Retourner à l'étape **1.**
7 **Fin Si**
8 **Retourner** (r, s)

•**Algorithme de vérification** : Toute personne ayant les informations publiques

$$\mathcal{K} = \{p, q, E, A, B\}$$

et une signature (prétendue) (r, s) de m peut vérifier si ce couple est une signature valide ou non en utilisant l'algorithme suivant :

Algorithme 30 : Algorithme de vérification

Entrées : Données publiques $\mathcal{K} = \{p, q, E, A, B\}$, le message m et la signature (r, s)

Sortie : Valide ou non valide

1 Calculer $w = s^{-1} \ (\bmod q)$;
2 Calculer $i = wH(m) \ (\bmod q)$; `/* H est une fonction de hachage cryptographique */`
3 Calculer $j = wr \ (\bmod q)$;
4 Calculer $iA + jB = (u, v)$;
5 **Si** $u \ (\bmod q) = r$ **Alors**
6 | **Retourner** *Valide*
7 **Sinon**
8 | **Retourner** *Invalide*
9 **Fin Si**

B.4 Courbes hyperelliptiques

Étant donné une courbe algébrique, nous pouvons former sa Jacobienne, le groupe de Picard de degré zéro. Nous savons que la Jacobienne d'une courbe elliptique est isomorphe à la courbe elliptique. Cependant, pour d'autres courbes, ce n'est pas le cas. Dans cette section, nous discutons des courbes hyperelliptiques, pour lesquelles la théorie peut être réalisée de façon assez explicite. Indépendamment, Koblitz [Kob87] et Muller [Mil86] ont suggéré l'utilisation des courbes hyperelliptiques pour construire des schémas cryptographiques. Nous discuterons de l'algorithme de Cantor, qui nous permet de calculer sur la

Jacobienne de courbes hyperelliptiques.

B.4.1 Définitions de base et propriétés

Dans la sous-section B.2.2, on définit la notion de diviseurs sur les courbes et le groupe de Picard. Nous discuterons également de ces notions dans le cas particulier des courbes hyperelliptiques.

Définition B.4.92 (courbe hyperelliptique)**.**

Une courbe hyperelliptique H de genre g $(g \geqslant 1)$ sur $\mathbb{K}$ est une courbe lisse donnée par une équation de la forme

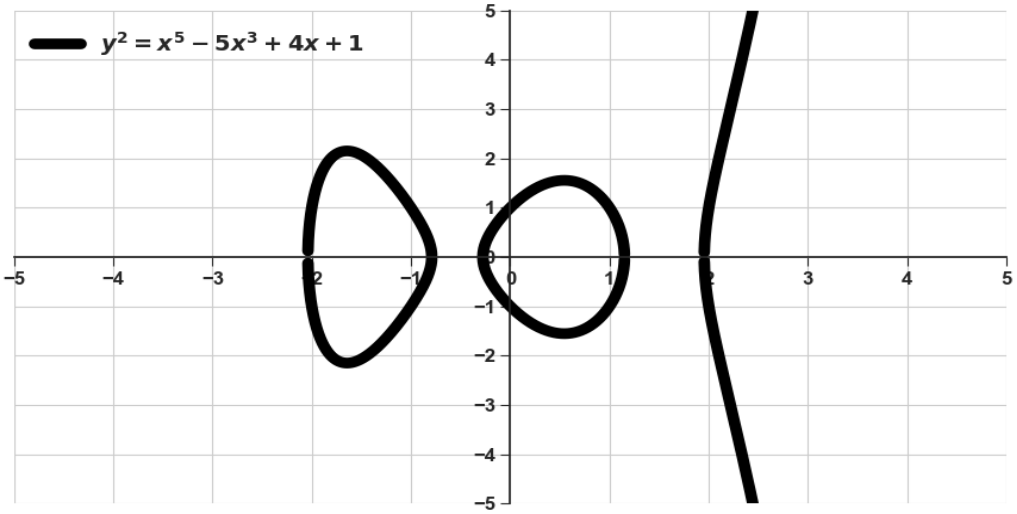
$$H : y^2 + h(x)y = f(x) \text{ in } \mathbb{K}[x, y] \tag{B.5}$$

où

— $h(x) \in \mathbb{K}[x]$ est un polynôme de degré au plus g,
— $f(x) \in \mathbb{K}[x]$ est un polynôme unitaire de degré $2g + 1$.

Remarque B.4.93
Dire que la courbe H est lisse signifie qu'il n'y a pas de solution $(x, y) \in \bar{\mathbb{K}} \times \bar{\mathbb{K}}$ qui satisfait simultanément l'équation $y^2 + h(x)y = f(x)$ et les deux équations de dérivée partielle $2y + h(x) = 0$ et $h'(x)y - f'(u) = 0$.

FIGURE B.8 – Courbe hyperelliptique : $y^2 = x^5 - 5x^3 + 4x + 1$ sur $\mathbb{R}$ (réalisé avec SimulaMath [9])



L'ensemble des points $\mathbb{K}$-rationnels sur H, noté $H(\mathbb{K})$, est l'ensemble de tous les points $(x, y) \in \mathbb{K} \times \mathbb{K}$ satisfaisant l'équation (B.5) avec le point à l'infini P_∞.

9. SimulaMath est un logiciel de calcul scientifique que nous avons conçu pour l'enseignement et l'apprentissage des mathématiques (voir chapitre 5).

Remarque B.4.94
Il existe également des courbes hyperelliptiques données par les équations de la forme (B.5) *avec* $\deg f = 2g + 2$.

Proposition B.4.95
Soit H une courbe hyperelliptique sur $\mathbb{K}$ définie par l'équation (B.5).

1. *Si $h(x) = 0$, alors* $\mathrm{char}(\mathbb{K}) \neq 2$
2. *Si* $\mathrm{char}(\mathbb{K}) \neq 2$ *alors le changement des variables* $(x, y) \to (x, y - h(x)/2)$ *transforme H sous la forme $y^2 = f(x)$ où* $\deg_x f = 2g + 1$
3. *Une courbe donnée par $C : y^2 = f(x)$ est une courbe hyperelliptique si et seulement* $\mathrm{char}(\mathbb{K}) \neq 2$ *et $f(x)$ n'a pas de racines multiples dans $\bar{\mathbb{K}}$.*

Preuve : Voir [MhWZ98].

□

Définition B.4.96 (points opposés, spéciaux et ordinaires)**.**

— Soit $P = (x, y)$ un point fini sur une courbe H. L'opposé de P est le point $\tilde{P} = (x, -y - h(x))$. On pose $\tilde{P_\infty} = P_\infty$.
— Si un point $P = (x, y)$ satisfait $\tilde{P} = P$ alors le point est dit spécial ; sinon, le point est dit ordinaire.

Définition B.4.97 (anneau des coordonnées)**.**

L'anneau des coordonnées de H sur $\mathbb{K}$, noté $\mathbb{K}[H]$, est l'anneau quotient $\mathbb{K}[H] = \mathbb{K}[x, y]/ < y^2 + h(x)y - f(x) >$ où $< y^2 + h(x)y - f(x) >$ désigne l'idéal sur $\mathbb{K}[x, y]$ engendré par le polynôme $y^2 + h(x)y - f(x)$.

Le polynôme $R(x, y) = y^2 + h(x)y - f(x)$ est irréductible sur $\bar{\mathbb{K}}$, et donc $\mathbb{K}[H]$ est un domaine intégral.

Définition B.4.98 (diviseur semi-réduit)**.**

Un diviseur semi-réduit est un diviseur de la forme

$$D = \sum n_i P_i - (\sum n_i) P_\infty$$

où chaque $n_i \geqslant 0$, $P_i \neq P_\infty$ et $P_i \neq \tilde{P}_j$ pour $i \neq j$.

Théorème B.4.99
Pour chaque $D \in \mathbb{D}^\circ$, il existe un diviseur semi-réduit $D_1 \in \mathbb{D}^\circ$ tel que $D_1 \sim D$.

Définition B.4.100 (diviseur réduit)**.**

Soit $D = \sum n_i P_i - (\sum n_i) P_\infty$ un diviseur semi-réduit. On dit que D est un diviseur réduit si $\sum n_i \leqslant g$.

Exemple B.4.6 *Pour une courbe hyperelliptique de genre $g = 2$, chaque diviseur réduit est équivalent à l'un des diviseurs suivants.*

1. $D_1 = P_1 - P_\infty$
2. $D_2 = P_1 + P_2 - 2P_\infty$
3. $D_3 = 2P_1 - 2P_\infty$

Théorème B.4.101
Soit H une courbe hyperelliptique de genre g donnée par $H : y^2 + h(x)y = f(x)$, où $h, f \in \mathbb{K}[x]$, $\deg f = 2g+1$, $\deg h \leqslant g$. Chaque classe de diviseurs non triviale sur $\mathbb{K}$ peut être représentée par une paire unique de polynômes $U(x)$ et $V(x), U, V \in \mathbb{K}[x]$ où

1. *$U(x)$ est unitaire ;*
2. $\deg_x V(x) < \deg_x U(x) \leqslant g$,
3. $U(x) | V(x)^2 + V(x)h(x) - f(x)$.

Preuve : Voir [CF06].

□

Définition B.4.102 (représentation de Mumford)**.**

La paire (U, V) est appelée la **représentation de Mumford** [Mum84] de la classe de diviseur correspondante. Dans de nombreuses situations, il est plus facile de travailler avec les représentations de Mumford que directement avec les classes de diviseurs.

Remarque B.4.103
Si $(U(x), V(x)) \in K[x] \times K[x]$ est la représentation de Mumford d'un diviseur réduit $D = \sum m_i P_i - (\sum m_i) P_\infty$ avec $P_i = (x_i, y_i)$ alors

1. $$U(x) = \prod (x - x_i)^{m_i}$$
   i.e le polynôme $U(x)$ décrit l'abscisse du support de D.
2. $$V(x_i) = y_i \text{ pour tout } i$$
   i.e le polynôme $V(x)$ interpole les ordonnées du support de D.
3. $\deg_x(V) < \sum m_i$,
4. $U(x) | V^2(x) + V(x)h(x) - f(x)$, *cette condition prend en compte les multiplicités m_i dans la définition de $U(x)$.*

Exemple B.4.7

Un diviseur $D = P - P_\infty$ avec $P = (u, v)$ a une représentation de Mumford $(x - u, v)$.

Considérons la courbe hyperelliptique H définie sur $\mathbb{R}$ par

$$H : y^2 = x^5 - 4x^4 - 14x^3 + 36x^2 + 45x = x(x + 1)(x - 3)(x + 3)(x - 5)$$

Les points $P_1 = (1, 8), P_2 = (3, 0)$ et $P_3 = (5, 0)$ appartiennent à $\mathbb{H}$. Considérons les diviseurs $D_1 = P_1 + P_2 - 2P_\infty$ et $D_2 = P_1 + P_3 - 2P_\infty$. Alors la représentation de Mumford de D_1 est $(a(x), b(x))$ avec $a(x) = (x - 1)(x - 3) = x^2 - 4x + 3$ et $b(x) = -4x + 12$ et D_2 est représenté par $(x^2 - 6x + 5, -2x + 10)$.

Théorème B.4.104 (Hasse-Weil)

Soit H une courbe hyperelliptique de genre g définie sur le corps fini $\mathbb{F}_q$. Alors on a : $\left(q^{n/2} - 1\right)^{2g} \leqslant \#\mathbb{J}\mathrm{ac}(H(\mathbb{F}_{q^n})) \leqslant \left(q^{n/2} + 1\right)^{2g}$ et $|\#H - (q + 1)| \leqslant 2g\sqrt{q}$

B.4.2 Algorithme de Cantor

La représentation de Mumford donne une représentation très concrète des points de J. Dans cette sous-section, nous présentons l'algorithme de Cantor (dû à David Cantor [Can87]) pour additionner des classes de diviseurs qui sont données par leur représentation de Mumford.

Remarque B.4.105 (notation)

*Soient $a, b \in K[x]$. Soit **CoefBezout**(a, b) les coefficients de Bezout e_1, e_2 de (a, b), i.e* $\gcd(a, b) = e_1 a + e_2 b$ *qui est équivalent à $e_1, e_2 =$ **CoefBezout**(a, b).*

Algorithme 31 : Algorithme de Cantor-Koblitz : Composition et réduction

Entrées : Deux diviseurs réduits $D_1 = [u_1(x), v_1(x)]$ et $D_2 = [u_2(x), v_2(x)]$ de la courbe $H : y^2 + h(x)y = f(x)$

Sortie : L'unique diviseur réduit $D = D_1 + D_2$

/* Composition */

$d_1 := \gcd(u_1, u_2)$;

$e_1, e_2 := \text{CoefBezout}(u_1, u_2)$; /* $e_1u_1 + e_2u_2 = \gcd(u_1, u_2)$ */

$d := \gcd(d_1, v_1 + v_2 + h)$;

$c_1, c_2 := \text{CoefBezout}(d_1, v_1 + v_2 + h)$; /* $c_1d_1 + c_2(v_1 + v_2 + h) = \gcd(d_1, v_1 + v_2)$ */

$s_1 := c_1e_1$;

$s_2 := c_1e_2$;

$s_3 := c_2$;

$u := \dfrac{u_1u_2}{d^2}$;

$v := \dfrac{s_1u_1v_2 + s_2u_2v_1 + s_3(v_1v_2 + f)}{d} \mod u$;

/* Réduction */

Faire

  $u' := \dfrac{f - v^2 - vh}{u}$;

  $v' := (-v - h) \mod (u')$;

  $u := u'$ et $v := v'$;

Tant que $\deg(u) > g$;

Rendre u unitaire;

Retourner $[u, v]$

L'algorithme de Cantor classique fonctionne dans un corps non binaire, c'est-à-dire pour des courbes hyperelliptiques de la forme $y^2 = f(x) \quad (h(x) = 0)$. Il a ensuite été étendu par Koblitz pour tout corps $\mathbb{K}$.

B.5 Implémentations

```
1  # -*- coding: utf-8 -*-
2
3
4  import random
5  import sympy
6  import sympy.abc
7
8
9  def inverse_modulaire(a, p):
10     assert sympy.isprime(p)
11     return gcdex(a, p) % p
12
13
14 class EllipticCurveFp():
```

```
15      """Courbe elliptique de la forme y^2=x^3+ax+b sur Fp p premier
16      """
17
18      def __init__(self, a=1, b=1, p=13, *args, **kwargs):
19          self.a = a % p
20          self.b = b % p
21          self.p = p
22          assert self.isEllipticCurve() and sympy.isprime(p) and p > 3
23          self.f = sympy.abc.x ** 3 + self.a * sympy.abc.x + self.b
24          self.POINT_INFINI = (None, None)
25
26      def __repr__(self):
27          texte = u"Courbe elliptique definie par : y^2 = {}".format(self.
                f)
28          texte += u" sur F_{}".format(self.p)
29          return texte.replace("**", "^")
30
31      def __eq__(self, courbe):
32          if self.a == courbe.a and self.b == courbe.b and self.p ==
                courbe.p:
33              return True
34          return False
35
36      def isEllipticCurve(self):
37          """ courbe de la forme E:y^2 = x^3 + a * x + b"""
38          a, b, mod = self.a, self.b, self.p
39          if (-16 * (4 * a ** 3 + 27 * b ** 2)) % mod != 0:
40              return True
41          return False
42
43      def jInvariantEllipticCurve(self):
44          """le j-invariant d'une  courbe elliptic de la forme E:y^2 = x^3
                + a * x + b"""
45          a, b, mod = self.a, self.b, self.p
46          j = 1728 * (4 * a ** 3) * inverse_modulaire(4 * a ** 3 + 27 * b
                ** 2, mod)
47          return j % self.p
48
49      def opposerPoint(self, P):
50          return P[0] % self.p, (-P[1]) % self.p
51
52      def addPoints(self, p1, p2):
53          """ Addition de P1 et P2 avec P1! = P2 != self.POINT_INFINI et
                P1 != -P2
54           dans E:y^2 = x^3 + a * x + b
55          """
56          mod = self.p
57          if p1 == self.POINT_INFINI or p2 == self.POINT_INFINI:
58              raise Exception()
59          else:
60              assert (p1[0] % mod, p1[1] % mod) != (p2[0] % mod, p2[1] %
                    mod)
61              assert (p1[0] % mod, p1[1] % mod) != self.opposerPoint(p2)
62              x1, y1 = p1
63              x2, y2 = p2
```

```
64              ld = (y2 - y1) * inverse_modulaire((x2 - x1) % mod, mod)
65              x3 = (ld ** 2 - x1 - x2) % mod
66              y3 = (ld * (x1 - x3) - y1) % mod
67              return (x3, y3)
68
69      def doublement(self, P):
70          """ Doublement de points dans E:y^2 = x^3 + a * x + b"""
71          a, mod = self.a, self.p
72          if P == self.POINT_INFINI or P[1] == 0:
73              return self.POINT_INFINI
74          else:
75              x1, y1 = P
76              ld = (3 * x1 ** 2 + a) * inverse_modulaire(2 * y1, mod)
77              x2 = (ld ** 2 - 2 * x1) % mod
78              y2 = (ld * (x1 - x2) - y1) % mod
79              return (x2, y2)
80
81      def additionPoints(self, p1, p2):
82          """Addition de P1 et P2 quelconques dans E:y^2 = x^3 + a * x + b
                """
83          mod = self.p
84          if p1 == self.POINT_INFINI and p2 == self.POINT_INFINI:
85              return self.POINT_INFINI
86          elif p1 == self.POINT_INFINI and p2 != self.POINT_INFINI:
87              return p2
88          elif p1 != self.POINT_INFINI and p2 == self.POINT_INFINI:
89              return p1
90          elif (p1[0] % mod, p1[1] % mod) == (p2[0] % mod, (p2[1]) % mod):
91              return self.doublement(p1)
92          elif (p1[0] % mod, p1[1] % mod) == self.opposerPoint(p2):
93              return self.POINT_INFINI
94          else:
95              return self.addPoints(p1, p2)
96
97      def multiplierPoint(self, P, k=2):
98          """Binary multiplication kP dans E:y^2 = x^3 + a * x + b"""
99          k = map(int, list(bin(k)[2:]))
100         L = len(k)
101         Q = self.POINT_INFINI
102         for j in range(L):
103             Q = self.doublement(Q)
104             if k[j] == 1:
105                 Q = self.additionPoints(Q, P)
106         return Q
107
108     def allCurvePoints(self):
109         """Tous les points de la courbe ellitique E:y^2 = x^3 + a * x +
                b"""
110         a, b, mod = self.a, self.b, self.p
111         f = lambda x: x ** 3 + a * x + b
112         Liste = [i for i in range(mod) if sympy.is_quad_residue(f(i),
                mod)]
113         L_points = ["Point_infini"]
114         for val in Liste:
115             L_points += [(val, sympy.sqrt_mod(f(val), mod)),
```

```
                            (val, (-sympy.sqrt_mod(f(val), mod)) % mod)]

        return set(L_points)

    def allPoints(self):
        L_points = self.allCurvePoints()
        L_points = map(str, list(set(L_points)))
        return "\n\t" + "\n\t".join(L_points)

    def estPoint(self, Q):
        if Q == self.POINT_INFINI:
            return True
        else:
            f = lambda x: x ** 3 + self.a * x + self.b
            return (Q[1] ** 2) % self.p == f(Q[0]) % self.p

    def ordre(self):
        return len(self.allCurvePoints())
```

```
# -*- coding: utf-8 -*-
"""
Cantor-Kobltz Algorithm for composition and reduction

@author: M. SECK
"""



def cantorAddDivisors(curve, D1, D2):
    f, h = curve.hyperelliptic_polynomials()
    u1, v1 = D1
    u2, v2 = D2
    d1, e1, e2 = xgcd(u1, u2)
    d, c1, c2 = xgcd(d1, v1 + v2 + h)
    s1, s2, s3 = c1*e1, c1*e2, c2
    u = u1*u2/(d^2)
    assert u.denominator() == 1
    u = u.numerator()

    v = (s1*u1*v2 + s2*u2*v1 + s3*(v1*v2 + f))/d
    assert v.denominator() == 1
    v = v.numerator().mod(u)

    u_prime = (f - v^2 - v*h)/u
    assert u_prime.denominator() == 1
    u_prime = u_prime.numerator()
    v_prime = (-v - h).mod(u_prime)

    while u_prime.degree() > curve.genus() :
        u_prime = (f - v_prime^2 - v_prime*h)/u_prime
        assert u_prime.denominator() == 1
        u_prime = u_prime.numerator()
        v_prime = (-v_prime - h).mod(u_prime)

    u_prime = (1/u_prime.leading_coefficient())*u_prime
```

```
37      return (u_prime, v_prime)
38
39  def mumford2formalSum(D):
40      u, v = D
41      points_d = []
42      for x_i, n_i in u.roots():
43          if not isinstance(v, Integer):
44              P = (x_i, v(x_i))
45          else:
46              P = (x_i, v)
47          points_d.extend([P for j in range(n_i) ])
48      return tuple(sorted(points_d))
49
50  # Examples
51
52  q = 11
53
54  k.<a> = GF(q); R.<x> = k[]
55  f=x^3-x-1
56  H = HyperellipticCurve(f=f)
57  D1 = (x-4, 2)
58  D2 = (x-2, 7)
59  cantorAddDivisors(H,D1,D2)
```

Table des figures

Liste des tableaux

Bibliographie

[AR14] G. Adj and F. Rodríguez-Henríquez. Square root computation over even extension fields. *IEEE Transactions on Computers*, 63(11) :2829–2841, 2014. 117, 119

[Atk92] A. Atkin. Probabilistic primality testing, summary by f. morain. research report 1779, inria, 1992. 117

[BBF+18] Nina Bindel, Jacqueline Brendel, Marc Fischlin, Brian Goncalves, and Douglas Stebila. Hybrid key encapsulation mechanisms and authenticated key exchange. Cryptology ePrint Archive, Report 2018/903, 2018. https://eprint.iacr.org/2018/903. 14

[BBJ+08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In Serge Vaudenay, editor, *Progress in Cryptology – AFRICACRYPT 2008*, pages 389–405, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 144, 145, 146

[BCI+10] Eric Brier, Jean-Sébastien Coron, Thomas Icart, David Madore, Hugues Randriam, and Mehdi Tibouchi. Efficient indifferentiable hashing into ordinary elliptic curves. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 237–254, 2010. 22, 23, 28, 31, 33, 66, 67, 69, 70, 71, 73, 77, 78, 79

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4) :235–265, 1997. Computational algebra and number theory (London, 1993). 86

[BDK+17] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals – kyber : a cca-secure module-lattice-based kem. Cryptology ePrint Archive, Report 2017/634, 2017. https://eprint.iacr.org/2017/634. 14

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '01, pages 213–229, London, UK, UK, 2001. Springer-Verlag. 11, 23, 28, 45, 66, 68, 77, 78, 79

[BHKL13] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator : elliptic-curve points indistinguishable from uniform random strings. In

Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, CCS '13, pages 967–980, New York, NY, USA, 2013. ACM. 20, 21, 22, 28, 30, 32, 45, 46, 47, 66, 70, 71, 72, 73, 77

[BL07] Daniel J. Bernstein and Tanja Lange. Faster addition and doubling on elliptic curves. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, pages 29–50, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 144

[BL14] Daniel J. Bernstein and Tanja Lange. Safecurves, 2014. 72

[BLMP19] Daniel J. Bernstein, Tanja Lange, Chloe Martindale, and Lorenz Panny. Quantum circuits for the csidh : Optimizing quantum evaluation of isogenies. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019*, pages 409–441, Cham, 2019. Springer International Publishing. 97

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security : Advances in Cryptology*, ASIACRYPT '01, pages 514–532, Berlin, Heidelberg, 2001. Springer-Verlag. 67, 77, 78

[BMP00] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *Proceedings of the 19th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'00, pages 156–171, Berlin, Heidelberg, 2000. Springer-Verlag. 77

[Can87] D. G. Cantor. Computing in the jacobian of a hyperelliptic curve. mathematics of computation, 48(177), 1987. 156

[CC03] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography : Public Key Cryptography*, PKC '03, pages 18–30, London, UK, UK, 2003. Springer-Verlag. 45

[CF06] H. Cohen and G. Frey. Handbook of elliptic and hyperelliptic curve cryptography, chapman & hall/crc taylor & francis group 6000 broken sound parkway nw, suite 300 boca raton, fl 33487-2742, 2006. 121, 131, 132, 155

[CS11] Abdoul Aziz Ciss and Djiby Sow. On a new generalization of huff curves. *IACR Cryptology ePrint Archive*, 2011 :580, 2011. 147

[DDK17] Nafissatou Diarra, Djiby, and Ahmed Youssef Ould Cheikh Khlil. Indifferentiable deterministic hashing into elliptic curves. *European Journal of Pure and Applied Mathematics*, 10(2) :363–391, 2017. 22, 23, 30, 66, 70, 71, 72, 74, 77, 79

[Den03] Alexander W. Dent. A designer's guide to kems. In Kenneth G. Paterson, editor, *Cryptography and Coding*, pages 133–151, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. 12, 14, 15

[Dev17] The Sage Developers. Sagemath, the sage mathematics software system (version 7.4), 2017. 5, 22, 44, 47, 61

[DGPS19] Wolfram Decker, Gert-Martin Greuel, Gerhard Pfister, and Hans Schönemann. SINGULAR 4-1-2 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de, 2019. 86

[DH06] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6) :644–654, September 2006. 10

[DJ11] Julien Devigne and Marc Joye. Binary huff curves. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, pages 340–355, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 147

[DSS18] Nafissatou Diarra, Michel Seck, and Djiby Sow. A note on encoding and hashing into elliptic and hyperelliptic curves, 2018. In A Collection of Papers in Mathematics and Related Sciences, a festschrift in honour of the late Galaye Dia (Editors : Seydi H., Lo G.S. and Diakhaby A.). Spas Editions, Euclid Series Book, pp. 565 –593. (2018) Doi : 10.16929/sbs/2018.100-07-01. 19

[Edw07] H.M. Edwards. A normal form for elliptic curves, bulletin of the american mathematical society 44, 2007. 144

[Eng01] Andreas Enge. Elliptic curves and their applications to cryptography, copyright © 1999 by kluwer academic publishers. second printing, 2001. 121, 133, 134

[Far11] Reza Rezaeian Farashahi. Hashing into hessian curves. In *Progress in Cryptology - AFRICACRYPT 2011 - 4th International Conference on Cryptology in Africa, Dakar, Senegal, July 5-7, 2011. Proceedings*, pages 278–289, 2011. 22, 66, 71, 77

[Far14] R. R. Farashahi. Hashing into hessian curves. applied cryptography, vol. 3, number 2, 2014. 28

[Feo17] Luca De Feo. Mathematics of isogeny based cryptography, 2017. 148

[FFS+13] Reza R. Farashahi, Pierre Alain Fouque, Igor E. Shparlinski, Mehdi Tibouchi, and J. Felipe Voloch. Indifferentiable deterministic hashing to elliptic and hyperelliptic curves. *Mathematics of Computation*, 82(281) :491–512, 2013. 19, 21, 23, 24, 28, 29, 30, 32, 40, 41, 43, 46, 66, 67, 73, 79, 80, 82, 83

[FJT13] Pierre-Alain Fouque, Antoine Joux, and Mehdi Tibouchi. Injective encodings to elliptic curves. In *Information Security and Privacy - 18th Australasian Conference, ACISP 2013, Brisbane, Australia, July 1-3, 2013. Proceedings*, pages 203–218, 2013. 28, 45, 66, 69, 70, 72, 73

[FNW10] Rongquan Feng, Menglong Nie, and Hongfeng Wu. Twisted jacobi intersections curves. In *Proceedings of the 7th Annual Conference on Theory and Applications of Models of Computation*, TAMC'10, pages 199–210, Berlin, Heidelberg, 2010. Springer-Verlag. 147

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology*, 26(1) :80–101, Jan 2013. 12, 15

[FT10] Pierre-Alain Fouque and Mehdi Tibouchi. Deterministic encoding and hashing to odd hyperelliptic curves. In *Pairing-Based Cryptography - Pairing 2010 -*

4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings, pages 265–277, 2010. 28, 72

[FT12] Pierre-Alain Fouque and Mehdi Tibouchi. Indifferentiable hashing to barreto-naehrig curves. In *Progress in Cryptology - LATINCRYPT 2012 - 2nd International Conference on Cryptology and Information Security in Latin America, Santiago, Chile, October 7-10, 2012. Proceedings*, pages 1–17, 2012. 22, 23, 66, 69, 70, 77, 79

[Gam85] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4) :469–472, 1985. 13, 15

[GAP19] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.10.2*, 2019. 86

[Gri07] P. A. Grillet. Abstract algebra (graduate texts in mathematics), second edition, 2007. 105, 108, 109, 111, 112, 113, 114

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING*, pages 212–219. ACM, 1996. 18

[Ham15] Mike Hamburg. Decaf : Eliminating cofactors through point compression. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology – CRYPTO 2015*, pages 705–723, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. 28

[Har77] Robin Hartshorne. *Algebraic Geometry*. Graduate Texts in Mathematics (Book 52). Springer ; 1st ed. 1977. Corr. 8th printing 1997 edition (April 1, 1997), 1977. 121, 122, 124, 130

[HBA+18] M. Hohenwarter, M. Borcherds, G. Ancsin, B. Bencze, M. Blossier, J. Éliás, K. Frank, L. Gál, A. Hofstätter, F. Jordan, Z. Konečný, Z. Kovács, E. Lettner, S. Lizelfelner, B. Parisse, C. Solyom-Gecse, C. Stadlbauer, and M. Tomaschko. GeoGebra 5.0.507.0, October 2018. http://www.geogebra.org. 86, 87

[Hes44] O. Hesse. Uber die elimination der variabeln aus drei algebrais-chen gleichungen vom zweiten grade mit zwei variabeln. journal für die reine und angewandte mathematik, 10, 1844. 147

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, pages 341–371, 2017. 12, 15

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru : A ring-based public key cryptosystem. In *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag, 1998. 13

[Huf48] G.B. Huff. Diophantine problems in geometry and elliptic ternary forms. duke math. j., 1948. 146

[Hun07] J. D. Hunter. Matplotlib : A 2d graphics environment. *Computing In Science & Engineering*, 9(3) :90–95, 2007. 86

[HWCD09] H. Hisil, K.H. Wong, G. Carter, and E. Dawson. Jacobi quartic curves revisited. in : Boyd c., gonzález nieto j. (eds) information security and privacy. acisp 2009. lecture notes in computer science,, 2009. 147

[HYW15] Xiaoyang He, Wei Yu, and Kunpeng Wang. Hashing into generalized huff curves. In *Information Security and Cryptology - 11th International Conference, Inscrypt 2015, Beijing, China, November 1-3, 2015, Revised Selected Papers*, pages 22–44, 2015. 22, 23, 46, 72, 73, 77, 79

[HYW17] Xiaoyang He, Wei Yu, and Kunpeng Wang. Hashing into twisted jacobi intersection curves. In *Information Security and Cryptology - 13th International Conference, Inscrypt 2017, Xi'an, China, November 3-5, 2017, Revised Selected Papers*, pages 117–138, 2017. 23, 24, 67, 75, 77, 79

[Ica09] Thomas Icart. How to hash into elliptic curves. *IACR Cryptology ePrint Archive*, 2009 :226, 2009. 28, 45, 66, 69, 70, 71

[Inc] Wolfram Research, Inc. Mathematica online, Version 12.0. Champaign, IL, 2019. 86

[Jab96] David P. Jablon. Strong password-only authenticated key exchange. *Computer Communication Review*, 26(5) :5–26, 1996. 77

[JOP+] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy : Open source scientific tools for Python, 2001–. 86

[JTV10] Marc Joye, Mehdi Tibouchi, and Damien Vergnaud. Huff's model for elliptic curves. In Guillaume Hanrot, François Morain, and Emmanuel Thomé, editors, *Algorithmic Number Theory*, pages 234–250, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 146, 147

[Kat10] Jonathan Katz. Digital signatures, 2010. Springer, DOI :10.1007/978-0-387-27712-7. 18

[KLR10] Jean-Gabriel Kammerer, Reynald Lercier, and Guénaël Renault. Encoding points on hyperelliptic curves over finite fields in deterministic polynomial time. In *Pairing-Based Cryptography - Pairing 2010 - 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings*, pages 278–297, 2010. 22, 23, 28, 32, 45, 66, 67, 71, 75, 77

[KM16] Neal Koblitz and Alfred Menezes. A riddle wrapped in an enigma. *IEEE Security and Privacy*, 14(6) :34–42, November 2016. 11, 18

[Kob87] N. Koblitz. Elliptic curve cryptosystems. mathematics of computation 48(177), 1987. 152

[Kob89] Neal Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1(3) :139–150, 1989. 45

[Koh96] D. R. Kohel. Endomorphism rings of elliptic curves over fnite felds [ph.d. tesis], university of california, berkeley, calif, usa, 1996. 149

[MAT18] Matlab optimization toolbox, 2018. 86

[MhWZ98] Alfred J. Menezes, Yi hong Wu, and Robert J. Zuccherato. An elementary introduction to hyperelliptic curves, 1998. 45, 121, 129, 154

[Mil86] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology*, CRYPTO '85, pages 417–426, Berlin, Heidelberg, 1986. Springer-Verlag. 152

[Mon87] P.L Montgomery. Speeding the pollard and elliptic curve methods of factorizations, math. comp, 1987. 142, 143

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Information Theory*, 39(5) :1639–1646, 1993. 68

[MRS+13] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik. *cluster : Cluster Analysis Basics and Extensions*, 2013. R package version 1.14.4 — For new features, see the 'Changelog' file (in the package source). 86

[MSP+17] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy : symbolic computing in python. *PeerJ Computer Science*, 3 :e103, January 2017. 86

[Mum84] David Mumford. Tata lectures on theta ii. birkhaüser, 1984. 155

[MVO96] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996. 9, 14

[Nist] National Institute of Standards and Technology (NIST) : Post-quantum cryptography : Nist's plan for the future. (2017). 15, 19

[OAYT19] Hiroshi Onuki, Yusuke Aikawa, Tsutomu Yamazaki, and Tsuyoshi Takagi. A faster constant-time algorithm of csidh keeping two points. Cryptology ePrint Archive, Report 2019/353, 2019. https://eprint.iacr.org/2019/353. 97

[OKS00] K. Okeya, H. Kurumatani, and K. Sakurai. Elliptic curves with the montgomery-form and their cryptographic applications. in : Imai h., zheng y. (eds) public key cryptography. pkc 2000. lecture notes in computer science, vol 1751. springer, berlin, heidelberg (2000), 2000. 143

[Oli] Travis Oliphant. NumPy : A guide to NumPy. USA : Trelgol Publishing, 2006–. [Online ; accessed <today>]. 86

[Ros95] Guido Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995. 86

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, February 1978. 13, 15

[RV11] P. Ramachandran and G. Varoquaux. Mayavi : 3d visualization of scientific data. *Computing in Science & Engineering*, 13(2) :40–51, 2011. 96

[S+09] W. A. Stein et al. *Sage Mathematics Software (Version 3.3)*. The Sage Development Team, 2009. `http://www.sagemath.org`. 86

[SBDK17] Michel Seck, Hortense Boudjou, Nafissatou Diarra, and Ahmed Youssef Ould Cheikh Khlil. On indifferentiable hashing into the jacobian of hyperelliptic curves of genus 2. In *Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings*, pages 205–222, 2017. 19, 23, 24, 46, 47, 60, 67, 75, 77, 79, 82

[SD18] Michel Seck and Nafissatou Diarra. Unified formulas for some deterministic almost-injective encodings into hyperelliptic curves. In *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*, pages 183–202, 2018. 19, 23, 24, 30, 66, 67, 75, 76, 77, 79, 80, 83

[Sec18] Michel Seck. Sage code for generalization of encodings into hyperelliptic curves, february 2018. available on github, 2018. 5, 22, 44, 61

[Sha72] D. Shanks. Five number-theoretic algorithms. proceedings of the second manitoba conference on numerical mathematics, 1972. 117, 118

[Sho94] P. W. Shor. Algorithms for quantum computation : Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, SFCS '94, pages 124–134, Washington, DC, USA, 1994. IEEE Computer Society. 18

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5) :1484–1509, October 1997. 18

[Sil09] J.H. Silverman. The arithmetic of elliptic curves, second edition, 2009. 121, 128, 129, 134, 149

[Ska05] M. Skalba. Points on elliptic curves over finite fields. *Acta Arith*, 117 :293–301, 2005. 68

[Sow16] Djiby Sow. Introduction à la sécurité prouvée en cryptographie à clés publique, 2016. Ecole CIMPA - Université des Technologies et de Médecine-Nouakchott 22-23 Février 2016. 16

[SV08] J. Scholten and F. Vercauteren. An introduction to elliptic and hyperelliptic curve cryptography and the ntru cryptosystem, 2008. 45

[SvdW06] Andrew Shallue and Christiaan van de Woestijne. Construction of rational points on elliptic curves over finite fields. In *Algorithmic Number Theory, 7th International Symposium, ANTS-VII, Berlin, Germany, July 23-28, 2006, Proceedings*, pages 510–524, 2006. 28, 45, 66, 68

[Tho00] Stephen Thomas. *SSL & TLS Essentials - Securing the Web*. Wiley Computer Publishing John Wiley & Sons, Inc., 2000. 9

[Tib11] M. Tibouchi. Hachage vers les courbes elliptiques et cryptanalyse de schémas rsa. thèse de doctorat de l'université paris-diderot-luxembourg. septembre, 2011. 28

[Ton91] A. Tonelli. Bemerkung uber die auflosung quadratischer congruenzen. gotinger nachrichten, 1891. 118

[Ula07] M. Ulas. Rational points on certain hyperelliptic curves over finite fields. *Bull. Pol. Acad. Sci. Math.*, 55(2) :97–104, 2007. 23, 28, 32, 45, 67, 75, 77

[Vel71] J. Velu. Isogénies entre courbes elliptiques, comptes rendus mathematique academie des sciences, paris, 1971. 20, 149

[VWY+13] Andreas Voellmy, Junchang Wang, Y Richard Yang, Bryan Ford, and Paul Hudak. Maple : Simplifying sdn programming using algorithmic policies. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 87–98, New York, NY, USA, 2013. ACM. 86

[Was03] Lawrence C. Washington. Elliptic curves : Number theory and cryptography, crc press, 2003. 121, 140, 141, 150, 151

[WF12] Hongfeng Wu and Rongquan Feng. Elliptic curves in huff's model. *Wuhan University Journal of Natural Sciences*, 17(6) :473–480, Dec 2012. 147

[YWL+15] Wei Yu, Kunpeng Wang, Bao Li, Xiaoyang He, and Song Tian. Hashing into jacobi quartic curves. In *Proceedings of the 18th International Conference on Information Security - Volume 9290*, ISC 2015, pages 355–375, New York, NY, USA, 2015. Springer-Verlag New York, Inc. 23, 66, 74, 77, 79

[YWL+16] Wei Yu, Kunpeng Wang, Bao Li, Xiaoyang He, and Song Tian. Deterministic encoding into twisted edwards curves. In *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, pages 285–297, 2016. 46, 66

[YWLT13] Wei Yu, Kunpeng Wang, Bao Li, and Song Tian. About hash into montgomery form elliptic curves. In *Information Security Practice and Experience - 9th International Conference, ISPEC 2013, Lanzhou, China, May 12-14, 2013. Proceedings*, pages 147–159, 2013. 23, 24, 66, 74, 77, 79

Université Cheikh Anta Diop de Dakar
Thèse de Doctorat Unique

Prénom et nom du candidat : Michel SECK

Titre : Sur la construction de fonctions de hachage sur les courbes (hyper)elliptiques

École Doctorale : Mathématiques et Informatique

Laboratoire : Laboratoire d'Algèbre de Cryptographie, de Géométrie Algébrique et Applications (LACGAA)

Date de soutenance : 2 Novembre 2020

Membres du jury

Président	:	Cheikh Thiécoumba GUEYE	Univ. Cheikh Anta Diop de Dakar	
Rapporteurs	:	Luca De Feo	Maître de Conférences (HDR)	IBM Research GmbH, Zurich
		Youssouf Mahamadou DIAGANA	Professeur Titulaire	Univ. Nangui Abrogoua, Abidjan
		Mamadou SANGHARÉ	Professeur Titulaire	Univ. Cheikh Anta Diop de Dakar
Examinateurs	:	Ismaïla DIOUF	Maître de Conférences (CAMES)	Univ. Cheikh Anta Diop de Dakar
		Amadou Lamine FALL	Maître de Conférences (CAMES)	Univ. Cheikh Anta Diop de Dakar
Directeur de thèse	:	Djiby SOW	Professeur Titulaire	Univ. Cheikh Anta Diop de Dakar

Résumé

❀ Pour la construction de protocoles ou de schémas en cryptographie basée sur les courbes elliptiques ou hyperelliptiques, il est parfois nécessaire de pouvoir représenter une chaîne de bits comme un point d'une courbe algébrique (encodage). La construction d'encodages et de fonctions de hachage sur les courbes a suscité une recherche active durant les deux dernières décennies. Comme contributions, nous avons réussi à construire des fonctions d'encodage déterministes et presque injectives sur la courbe hyperelliptique $\mathbb{H}_g : y^2 = f_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0$ pour $g = 1, 2, \ldots, 9$. Nous avons aussi montré comment on peut construire des fonctions de hachage ayant de bonnes propriétés cryptographiques sur ces courbes, et ce, de façon déterministe en utilisant nos encodages et quelques résultats de Farashahi *et al.* (Mathematics of Computation, 2013).

❀ Nous décrivons également SimulaMath, un logiciel de calcul scientifique pour la recherche, l'apprentissage et l'enseignement en mathématiques pour presque tous les niveaux (collège, lycée et université). Il couvre de nombreux domaines des mathématiques comme l'algèbre linéaire, l'analyse, la théorie des nombres, la statistique descriptive, les distributions de probabilités, les graphiques en 2D et 3D, les bases de Groebner, les réseaux arithmétiques, les courbes elliptiques et les codes linéaires.

Abstract

❀ For the construction of cryptographic protocols or schemes in cryptography based on elliptic or hyperelliptic curves, it is sometimes necessary to be able to represent a string of bits as a point on an algebraic curve (encoding). The construction of encodings and hash functions on curves has been the subject of an active research during the last two decades. As contributions, we have successfully constructed deterministic and almost injective encoding functions on the hyperelliptic curve $\mathbb{H}_g : y^2 = f_g(x) = x^{(2g+1)} + a_{(2g-1)}x^{(2g-1)} + a_{(2g-3)}x^{(2g-3)} + \ldots + a_1x + a_0$ for $g = 1, 2, \ldots, 9$. We have also shown how one can construct hash functions with good cryptographic properties on these curves in a deterministic way using our encodings and some results of Farashahi *et al.* (Mathematics of Computation, 2013).

❀ We also describe SimulaMath, a scientific computation software for research, learning and teaching mathematics for almost all levels (middle school, high school and university). It covers many areas of mathematics such as linear algebra, calculus, number theory, descriptive statistics, probability distributions, 2D and 3D graphics, Groebner bases, lattices, elliptic curves, and linear codes.