Pairings in cryptology: efficiency, security, and applications.

M. Charlemagne

under the supervision of Professor M. Scott

School of Computing

Dublin City University

A thesis submitted for the degree of

PhilosophiæDoctor in Computer Science

December 2010

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of PhilosophiæDoctor in Computer Science is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: (Candidate) ID No.: Date:

To. . . nobody.

Acknowledgements

In order to do justice to the reader I would like to start by thanking him for reading this thesis, as it also means that I did not write it in vain. Nevertheless he should note that all this work would not have been possible without the help of many others.

First and foremost I want to thank my supervisor Mike Scott, whose availability, enthusiasm and knowledge have been of a major help throughout those three first years of research. His guidance together with the help and comments from Robert Ganger, David Freeman, Jacques Patarin, Laura Hitt, Alice Silverberg, Paulo Barreto, Kim Hyun Sung and Gary McGuire, greatly helped me to overcome all the small mistakes and misunderstandings that punctuate a learning process. Therefore to each one of them I want to say thank you.

I would also like to thank François Morain, the algorithmic number theory group in Bordeaux, especially Andreas Enge and Jean-Marc Couveignes, and Matthieu Bontrond for their helpful insights on the variate areas that I have investigated throughout my work.

Beyond the scope of this work, I also want to thank Naomi Benger, Luis Dominguez, Ezechiel Kachisa, Yu Chen and Denis Butin for all the time we

spent together, and the chats we had over the years in the office.

A PhD features different stages including studying, understanding and discovering new things, unfortunately it also implies the tedious task of writing a thesis; and once it is written it has to be proof-read... In this regard I really want to thank Peter and Summer for all the time and effort they put in this even more boring task.

Finally, I would like to acknowledge the thousands of individuals who have coded for the LaTeXproject, allowing me to write this thesis and you to hopefully enjoy it...

Contents

List of Algorithms

List of Figures

List of Tables

Notation

$[\mathbf{L} : \mathbf{K}]$	The degree of an extension $\mathbf{L}$ over a field $\mathbf{K}$
(p)	The ideal generated by (p)
$\mathbb{C}$	The field of complex numbers
D	The CM discriminant of an ordinary pairing-friendly elliptic curve $E/\mathbf{K}$
$\mathrm{Div}(R)$	The divisor group of R
$\mathrm{div}(\cdot)$	The divisor of an element of a ring R
$\mathrm{Deg}(\cdot)$	The degree of a divisor
$\deg(\cdot)$	The degree of a polynomial
Δ	The discriminant of a curve C
$E/\mathbf{K}$	The elliptic curve E defined over a field $\mathbf{K}$
$E(\mathbf{K})$	The group of points on the elliptic curve $E/\mathbf{K}$
$E'/\overline{\mathbf{K}}$	A twist of an elliptic curve $E/\mathbf{K}$
$e(\cdot,\cdot)$	A pairing map
$\mathbb{F}_q$	A finite field, with $q = p^m$, where p is prime and m is a positive integer
$\varphi(\cdot)$	The Euler totient function
$\Phi_k(\cdot)$	The kth cyclotomic polynomial
$\phi^*(\cdot)$	The dual isogeny of ϕ
$\mathrm{Hom}(\mathbb{G}_1, \mathbb{G}_2)$	The abelian group consisting of all group homomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$
$J(\mathbf{K})$	The Jacobian of a curve C
g	The genus of a curve c
$J(\mathbf{K})[r]$	The r-torsion subgroup of $J(\mathbf{K})$

$\overline{\mathbf{K}}$	The algebraic closure of a field $\mathbf{K}$
$\mathbf{K}[x]$	The polynomial ring of a field $\mathbf{K}$
$\mathbf{K}(x)$	The field of rational functions of a field $\mathbf{K}$
k	The embedding degree of a pairing-friendly elliptic curve E/F_q
$L_q(\cdot,\cdot)$	The L-notation to express subexponential complexity
μ_r	The group of the rth roots of unity
$\mathbb{N}$	The set of natural numbers
$O(\cdot)$	The big O notation
$\mathcal{O}$	The point at infinity of an elliptic curve E
$\mathfrak{O}_{\mathbf{K}}$	The ring of integers of a field $\mathbf{K}$
$\mathbb{P}^1$	The projective line
$\mathbb{P}^2$	The projective plane
$\pi_p(\cdot)$	The p-power Frobenius
$\mathbb{Q}$	The field of rational numbers
$\mathbb{R}$	The field of real numbers
$\overline{R}$	The integral closure of a ring R
ρ	The ρ-value of a pairing-friendly elliptic curve E/F_q
$\mathbb{Z}$	The ring of integers

Abstract

The study of pairings can be considered in so many different ways that it may not be useless to state in a few words the plan which has been adopted, and the chief objects at which it has aimed. This is not an attempt to write the whole history of the pairings in cryptology, or to detail every discovery, but rather a general presentation motivated by the two main requirements in cryptology; efficiency and security.

Starting from the basic underlying mathematics, pairing maps are constructed and a major security issue related to the question of the minimal embedding field [12][1] is resolved. This is followed by an exposition on how to compute efficiently the final exponentiation occurring in the calculation of a pairing [124][2] and a thorough survey on the security of the discrete logarithm problem from both theoretical and implementational perspectives. These two crucial cryptologic requirements being fulfilled an identity based encryption scheme taking advantage of pairings [24][3] is introduced. Then, perceiving the need to hash identities to points on a pairing-friendly elliptic curve in the more general context of identity based cryptography, a new technique to efficiently solve this practical issue is exhibited [123][4].

[1] Joint work with N. Benger and D. Mandell Freeman.
[2] Joint work with M. Scott, N. Benger, L. Dominguez and E. Kachisa.
[3] Joint work with Y. Chen, Z. Guan, J. Hu and Z. Chen.
[4] Joint work with M. Scott, N. Benger, L. Dominguez and E. Kachisa.

Unveiling pairings in cryptology involves a good understanding of both mathematical and cryptologic principles. Therefore, although first presented from an abstract mathematical viewpoint, pairings are then studied from a more practical perspective, slowly drifting away toward cryptologic applications.

1

Introduction

> All things that are still to come lie in uncertainty; live straightway!
>
> L. Seneca

In his essay *A mathematician's apology* [63], Hardy justifies that mathematics should be pursued for its own sake, arguing that, due to its abstract nature, its very uselessness means no potential misuse to cause harm. At the end of World War II, this view was shared by other mathematicians, who did not want their work to lead to new destructive weapons. Therefore, some researchers refocused their work on less applicable mathematics like number theory or algebraic geometry. In a sense, they implicitly agreed with Hartmanis' observation that theoretical results that are hard to prove are useless in practice [78]. In this regard number theory is a very interesting area to investigate, although as one of its goals for some researchers in the 20th century was not to have any application, it has become more than half a century later an integral part of our lives through its use in *cryptology*.

The etymology of the word cryptology clearly relates to "the science of secrets". Hence, as a matter of fact it is not something new, but rather an ancient science, that faced a real revolution only few decades ago, with the introduction of mathematics at

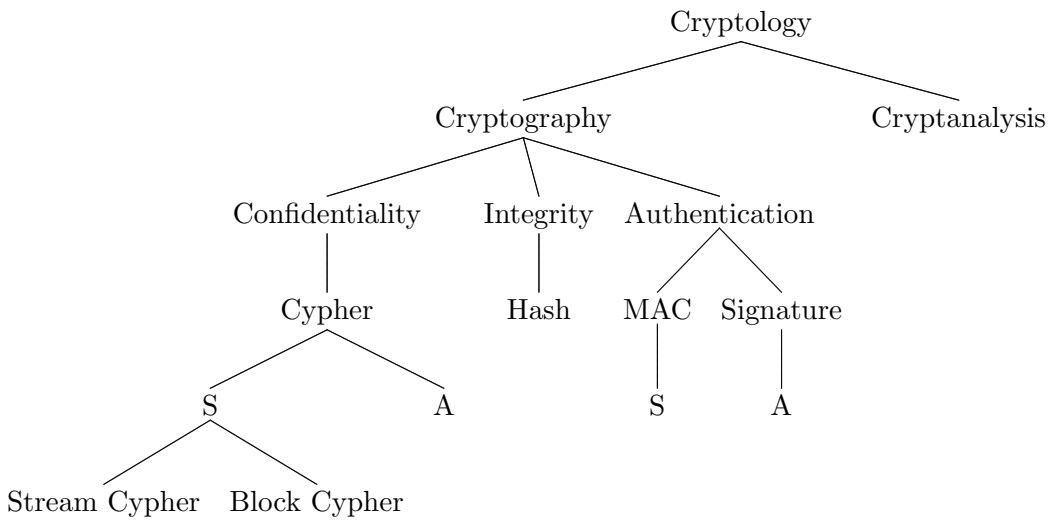

Figure 1.1: Modern cryptology

its core.

Hiding information can be achieved by using a secret, that is only known to the ones allowed to access some given data. This secret, or *key*, can be of two natures: symmetric or asymmetric, depending on whether or not the secret used to hide, or *encrypt*, the data is the same which must be used to reveal, or *decrypt* the information. Denoting symmetric and asymmetric by the letter S and A, respectively, figure 1.1 gives a picture of how cryptology splits into varied subfields.

The links between mathematics and cryptology appeared during the 1970s, with the introduction of asymmetric cryptography. In fact, thanks to mathematical problems hard to solve but that can easily be constructed in practice, it became possible for anybody to encrypt a message using a *public key*, known to anyone, and such that only the owner of the corresponding *private key* could decrypt it. One of the most important and intensively studied such problem for the past twenty years is the Discrete Logarithm Problem (DLP).

From a mathematical perspective it consists in finding the logarithm of a number

β, given a generator α of a finite field $\mathbb{F}$. More formally this is stated in the problem below:

Problem 1 (DLP). *Let $\mathbb{F}_q$ be a finite field of characteristic p, with $q = p^n$, p a prime and n a positive integer. Let α be a generator of $\mathbb{G}$, a subgroup of $\mathbb{F}_q$. For a given $\beta \in \mathbb{G}$, find x such that $\beta = \alpha^x \mod q$.*

From a cryptographic point of view this problem gives rise to a *one-way function*, a function easy to compute in one direction, the exponentiation side, but difficult to compute in the opposite direction, the inverse of the exponentiation, the DLP.

Although the DLP is extensively used in cryptography we only present the Diffie-Hellman key exchange protocol [31] as an example on how to take advantage of a one-way function. If **A** and **B** want to start an encrypted discussion then, obviously they need to share a secret, only known to them. Assuming that **A** and **B** agree on a finite field $\mathbb{F}_q$ and a generator α of a subgroup $\mathbb{G} \subset \mathbb{F}_q$, and that they both possess a secret x_a and x_b, respectively, Diffie and Hellman proposed the following solution to solve their key exchange problem.

- Initial state:

  A: $\mathbb{F}_q$, α, x_a **B:** $\mathbb{F}_q$, α, x_b

- **A** sends α^{x_a} to **B** and **B** sends α^{x_b} to **A**:

  A: $\mathbb{F}_q$, α, x_a, α^{x_b} **B:** $\mathbb{F}_q$, α, x_b, α^{x_a}

- **A** and **B** compute $(\alpha^{x_b})^{x_a}$ and $(\alpha^{x_a})^{x_b}$, respectively:

  A: $\mathbb{F}_q$, α, x_a, $\alpha^{x_b \cdot x_a}$ **B:** $\mathbb{F}_q$, α, x_b, $\alpha^{x_a \cdot x_b}$

- **A** and **B** have a common secret key $\alpha^{x_a \cdot x_b}$:

  A, B: $\mathbb{F}_q$, α, $\alpha^{x_a \cdot x_b}$

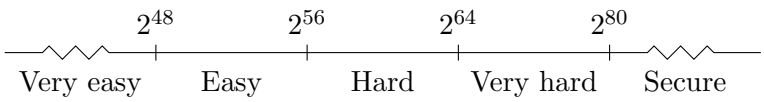

Figure 1.2: Security levels

In this protocol clearly **A** and **B** cannot recover any information relative to the secret key of one another without solving the DLP. This small example steers our attention to two main principals on which cryptography relies: security and efficiency.

Although obvious, the first one needs further refinements, while at first glance the second one may not be important. In fact, security is totally relative to the importance of the data. Clearly if **A** tells **B** in an encrypted conversation that he is leaving tomorrow, there is no need for this message to be secure over a day. Therefore, when using cryptography the first thing to know is why we do require cryptography, and then assess which security level should be matched. Taking into account the best actual computational power available, a hard problem is considered secure when the best algorithm to solve it has complexity at least 2^{80}, or even 2^{128} as some argue [101]. Figure 1.2 clarifies how hard it is to solve a problem featuring a given complexity. In the case mentioned above a security of 2^{56} would be more than sufficient, however when considering the general case we refer to 2^{80} as the standard security level.

In the case of the Diffie Hellman key exchange protocol, it is required for the standard security level that $\mathbb{F}_q$ have a size expressed by a 1024 bit integer, which in turn, implies dealing with very large numbers and thus a lower efficiency. In this regard a breakthrough was made by Miller [94] and Koblitz [77] when they introduced the Elliptic Curve Discrete Logarithm Problem (ECDLP), the equivalent of the DLP on an elliptic curve.

Problem 2 (ECDLP). *Let E be an elliptic curve over $\mathbb{F}_q$, and let P be a generator of a subgroup $\mathbb{G}$ of $E(\mathbb{F}_q)$. For a given point $Q \in G$, find the integer x such that $[x]P = Q$.*

This new problem allows shorter keys in cryptographic protocols as the available algorithms to solve the ECDLP are a lot slower than the one to solve the DLP. For instance, to achieve the same standard security level the key is only required to be 160bits long when elliptic curves are used, compared to the 1024bits required over a finite field.

From this observation, in 1993, Menezes, Okamoto, and Vanstone (MOV) [91] had the idea to transform the ECDLP into the DLP using a map called *pairing*, weakening protocols based on the hardness of the ECDLP. However this attack applied only to elliptic curves that fail the MOV test, that is have a small embedding degree. Then, at the end of the nineties two other main attacks were published. The first one, which focuses on the use of Weil-Descent to map the ECDLP to the DLP on hyper-elliptic curves, is due to Galbraith and Smart [50], while the second one, independently discovered by Smart [128] and Satoh and Araki [117], targets elliptic curves of trace one. A few years later, pairings have been looked at from a different angle. Instead of using it destructively such as in the MOV case, one took advantage of some of its properties in order to construct new protocols.

More formally, a pairing is a map, e from additive groups $\mathbb{G}_1$ and $\mathbb{G}_2$ into a multiplicative group $\mathbb{G}_T$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. A pairing has three different variants depending on the groups $\mathbb{G}_1$ and $\mathbb{G}_2$:

- *Type I:* $\mathbb{G}_1 = \mathbb{G}_2$
- *Type II:* $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficiently computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$
- *Type III:* $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists no efficiently computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$

For some given $P_1,\ P_2,\ P \in \mathbb{G}_1$ and $Q_1,\ Q_2,\ Q \in \mathbb{G}_2$ a pairing has the following properties:

- *Bilinearity*:

$$\begin{aligned} e(P, Q_1 + Q_2) &= e(P, Q_1)e(P, Q_2) \\ e(P_1 + P_2, Q) &= e(P_1, Q)e(P_2, Q), \end{aligned}$$

- *Non-degeneracy*:

$$\begin{aligned} &\forall P \in G_1,\ P \neq \infty \quad \exists Q \in G_2 \text{ such that } e(P, Q) \neq 1 \\ &\forall Q \in G_2,\ Q \neq \infty \quad \exists P \in G_1 \text{ such that } e(P, Q) \neq 1, \end{aligned}$$

- e is efficiently computable.

The most useful property of a pairing is its bilinearity which allows some new constructions as the one proposed by Joux [72] to handle a tripartite key exchange. Assuming that **A**, **B** and **C** agreed on an elliptic curve over $\mathbb{F}_q$, a generator P of a subgroup of $E(\mathbb{F}_q)$, a type I pairing and that each one owns a private key x_a, x_b and x_c, respectively, it works as follows.

- Initial state:

A: $E(\mathbb{F}_q)$, P, x_a **B:** $E(\mathbb{F}_q)$, P, x_b **C:** $E(\mathbb{F}_q)$, P, x_c

- **A**, **B** and **C** broadcast $Q_a = [x_a]P$, $Q_b = [x_b]P$ and $Q_c = [x_c]P$, respectively:

A: $e(Q_b, Q_c)^{x_a}$ **B:** $e(Q_a, Q_c)^{x_b}$ **C:** $e(Q_a, Q_b)^{x_c}$

- **A**, **B** and **C** share the same secret key:

A, B, C: $e([x_a.x_b.x_c]P, P) = e(P, P)^{x_a x_b x_c}$

As in many identity based encryption schemes [17], this protocol relies on the bilinear property of the pairing and the hardness of both the ECDLP and the DLP to

be solved. As such it is of a vital importance to consider, both the security and the efficiency of the pairings in order to be able to safely use them in the field of pairing based cryptography.

Our study starts with a high level overview of the mathematics involved so that the reader can grasp the underlying theory in which algebra, number theory, and algebraic geometry are tightly entangled (chapter 2). This includes many results that were thought useless and completely abstract when discovered. However, with the introduction of mathematics into the cryptographic world, they became very helpful in the construction of new secure protocols and as such, are now used in day to day life, for example in computers, cell phones, credit cards etc...

Next, we will survey curves suitable for a pairing, and give an algorithm to compute the pairing of two given points on an elliptic curve (chapter 3). Then, we present some criteria to ensure that pairings map to the expected field, i.e. not a subfield of the targeted field (chapter 4).

The next stage, is to consider the two main requirements of cryptography, that are the efficiency of the computation of the pairings (chapter 5) and their security (chapter 6). This permits the construction of a new identity based encryption scheme by taking advantage of the bilinear property of the pairings (chapter 7), which in turn leads to the more general question of knowing how to efficiently hash identities to points on a curve (chapter 8).

2

Mathematics and pairings

> If I didn't understand anything to mathematics I would be ashame to say it; letting others know that you are an idiot is not the best way to present yourself.
>
> B. Vian

Although pairings can be viewed from a simple angle as basic bilinear maps, a lot more is involved when one wants to present them from a more mathematical perspective, implying a good understanding of the underlying structures. Therefore, through this chapter we will present some important results, on the cross roads of number theory, algebra and algebraic geometry, always keeping in mind our initial target, pairings.

2.1 Rings and ideals

One of the main goal of mathematics is to classify objects which exhibit common properties. When a common property is found among a few *objects* they form a set called a *class*. However looking only at sets of objects does not allow for a satisfying classification as for example the set of all sets cannot be defined... This leads to the more general definition of *category*, giving rise to the notion of *morphism* or *map*.

Together with some maps, sets can take special algebraic structures. For example, $\mathbf{K} = (\mathbb{Q}, +, .)$ defines a *field* with *integer ring* $\mathfrak{O}_{\mathbf{K}} = (\mathbb{Z}, +, .)$. This ring has the property of being a *unique factorization domain*. However, when looking at *number fields*, extensions of $\mathbb{Q}$, this property is often lost, as for instance in the case of $\mathbf{K} = \mathbb{Q}(\sqrt{-5})$ and $\mathfrak{O}_{\mathbf{K}} = \mathbb{Z} + \mathbb{Z}\sqrt{-5}$ where 21 can de factorised into prime factors using the two following decompositions:

$$\begin{aligned} 21 &= 7.3 \\ 21 &= (1 + 2\sqrt{-5})(1 - 2\sqrt{-5}). \end{aligned}$$

To overcome this failure, Kummer had the idea of finding a way to embed the integers of $\mathbf{K}$ into a "larger domain" made of "ideal numbers" which would factorise into primes. Following the preceding example, and using $\mathfrak{p}_i$ to denote those "ideal numbers", would lead to:

$$7 = \mathfrak{p}_1\mathfrak{p}_2, \quad 3 = \mathfrak{p}_3\mathfrak{p}_4, \quad (1 + 2\sqrt{-5}) = \mathfrak{p}_1\mathfrak{p}_3, \quad (1 - 2\sqrt{-5}) = \mathfrak{p}_2\mathfrak{p}_4.$$

And then the unique factorisation appears again as $21 = (\mathfrak{p}_1\mathfrak{p}_2)(\mathfrak{p}_3\mathfrak{p}_4) = (\mathfrak{p}_1\mathfrak{p}_3)(\mathfrak{p}_2\mathfrak{p}_4)$.

Later on, Dedekind [36] extended this idea by re-introducing the *ideals* in a way that allowed him to define the division of a by b as the inclusion on b in the ideal generated by a. Hence using this abstraction it was possible to reintroduce a unique factorisation on the ring of integers of a number field. In fact this gave rise to the notion of ideal factorisation. A ring such that every nonzero proper ideal factors into a unique product of primes ideals is called a *Dedekind domain*. The following theorem gives more details on the factorisation of ideals over such rings.

Theorem 2.1.1. ([116]) Let R be a Dedekind domain with quotient field $\mathbf{K}$. Let $\mathbf{L}$ be a finite extension of degree n of $\mathbf{K}$ and B the integral closure of R in $\mathbf{L}$. Then B is

again a Dedekind domain and every non-zero prime ideal $\mathfrak{p}$ of the ring R decomposes in B in a unique way into a product of prime ideals:

$$\mathfrak{p}B = \prod_i \mathfrak{p}_i^{e_{\mathfrak{p}_i}}$$

The $\mathfrak{p}_i$ are precisely those prime ideals $\mathfrak{P}$ of B such that $\mathfrak{p} = \mathfrak{P} \bigcap R$. $e_{\mathfrak{p}_i}$ is called the ramification index in $\mathfrak{p}_i$ and the degree of the field extension $f_i = [B/\mathfrak{p}_i : B/\mathfrak{p}]$ is called the inertia degree of $\mathfrak{p}_i$ over $\mathfrak{p}$.
In the case of $\mathbf{L}$ being a separable extension, $\sum_i e_{\mathfrak{p}_i} f_{\mathfrak{p}_i} = n$.

Example 2.1.2. ([103]) We want to study the ramification points in the following case: let $R = \mathbb{R}[X], \quad \mathbf{K} = \mathbb{R}(X), \quad \mathbf{L} = \mathbf{K}[Y]/\langle Y^2 - X\rangle$.
First by remarking that $\mathbb{R}[X]$ is principal, as $\mathbb{R}$ is a field, we see that all the prime ideals of R are either defined by polynomials of degree 1, or irreducible polynomials of degree 2 with discriminant less than zero. Thus 4 different cases occur:

1. $\mathfrak{p} = (X - \lambda), \lambda \in \mathbb{R}^{*+}$. In this case $Y^2 - \lambda = (Y - \sqrt{\lambda})(Y + \sqrt{\lambda})$, and then both $(Y - \sqrt{\lambda})$ and $(Y + \sqrt{\lambda})$ have ramification index and inertia degree equal to 1.

2. $\mathfrak{p} = (X - \lambda), \lambda \in \mathbb{R}^{*-}$. As $\mathfrak{p}$ cannot be factorised, so it has ramification index equal to 1 and inertia degree equal to 2.

3. $\mathfrak{p} = (X^2 + aX + b), a, b \in \mathbb{R}$, with discriminant less than zero. By rewriting $\mathfrak{p} = (X - z)(X - \overline{z}) = (Y^2 - z)(Y^2 - \overline{z})$, we get 4 complex points, $\sqrt{z}, \overline{\sqrt{z}}, -\sqrt{z}, -\overline{\sqrt{z}}$, considered as only 2 real points, yielding, for each point, ramification index 1 and inertia degree 1.

4. $\mathfrak{p} = (Y^2)$. In this case the ramification index is 2 and the inertia degree is 1.

We note that, as $\mathbf{L}$ is a separable extension of degree 2, in each case there is $\sum_i e_{\mathfrak{p}_i} f_{\mathfrak{p}_i} = 2$.

Ramification and algebraic structures can also be looked at from a geometrical perspective, by seeing numbers as functions over a topological space. To do so, we define the *spectrum* of a ring R, as $X = Spec(R)$, to be the set of all maximal ideals $\mathfrak{p}$ in R, and we consider the sets $V(\mathfrak{b}) = \{\mathfrak{p}/\mathfrak{p} \supseteq \mathfrak{b}\}$, for $\mathfrak{b}$ an ideal, to be closed. X becomes a topological space, endowed with *Zariski topology* [37]. It can be made slightly more general by extending the set of close sets to all prime ideals, not only the maximal ones, and adding a generic point (0), which closure is the whole space X.

Example 2.1.3. ([103]) Looking at example 2.1.2 from a geometrical perspective, the question is to know how the points of the curve $Y^2 = X$ ramify over the $R = \mathbb{R}[X]$. Using Zariski topology, the prime ideals of R are the closed points of $Spec(R)$, and, as such, are precisely the points defined by the 4 different types of ideals listed above. In figure 2.1, the line represents A, and the curve has equation $Y^2 = X$. It becomes clear that the fibers of the points in R, that is the spectrum of an algebra of dimension 2 over $\mathbb{R}$, consist of 2 points, with residue field $\mathbb{R}$, when $\lambda > 0$ and one point with residue field $\mathbb{C}$ if $\lambda < 0$. When the fiber has only one point, i.e. $\lambda = 0$ here, the map $Y^2 - X$ is said to be ramified, and has ramification index equals 2. The special case of ideals generated by an irreducible polynomial of degree 2 leads to 2 complex points z and $\overline{z}$ considered as a unique real point $(z, \overline{z})$, with residue field $\mathbb{C}$.

Although Dedekind domains have a really nice structure, it may happen that the rings we consider are not integrally closed and as such cannot be Dedekind domains but only *noetherian*. When such rings have *Krull dimension* less or equal to 1, i.e. such that the length of the longest, strictly increasing, chain of prime ideals of R is not larger than 1, it is still possible to overcome the situation using the following result.

Theorem 2.1.4 (Krull-Akizuki [100, chapter 1, proposition 12.8]). Let R be a noetherian integral domain of Krull dimension less or equal to 1, $\mathbf{K}$ its fraction field and $\mathbf{L}$ an extension of finite degree. Then, every ring B such that $R \subset B \subset \mathbf{L}$ is

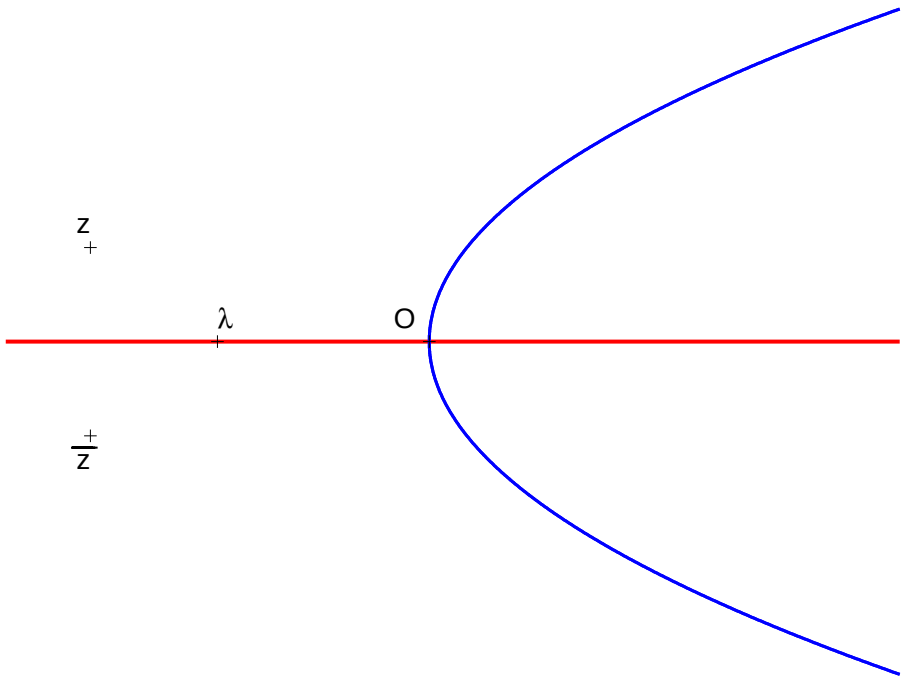

Figure 2.1: Geometrical interpretation of example 2.1.2.

noetherian of dimension less or equal than 1.

By applying this theorem to $\mathbf{K} = \mathbf{L}$, we see that the integral closure of R is a Dedekind domain. This means that given a noetherian integral domain it is possible to benefit from the structure of a Dedekind domain by taking its integral closure. From a geometrical point of view it means that given a singular curve it can be rendered smooth by looking at its integral closure. This process is called *resolution* of singularities.

Until now we have only given a rough presentation of the links between algebra and geometry, so we will now have a proper look at what is a curve, how it is defined and how to classify them. Table 2.1 from [103] gives a correspondence between the most common terms of algebra and algebraic geometry.

Algebra	Algebraic geometry
Ring	Affine scheme
R	$Spec(R)$
Algebra over $\mathbf{K}$	Affine scheme over $\mathbf{K}$
Finite algebra over $\mathbf{K}$	Algebraic variety over $\mathbf{K}$
Noetherian over $\mathbf{K}$ and reduced	Reduced algebraic variety over $\mathbf{K}$
Prime ideal $\mathfrak{p}$	Point P
Localisation in $\mathfrak{p}$	Neighborhood of P
A unique minimal prime ideal	Irreducible
Integral	Reduced and irreducible
Localisation in each prime ideal is integrally closed	Normal
Integrally closed	Reduced, irreducible and normal
Krull dimension 1	Algebraic variety of dimension 1

Table 2.1: Correspondence table between algebra and algebraic geometry

2.2 Varieties and curves

The main idea to link algebra and geometry is to find a one-to-one correspondences between algebraic and geometrical structures. To do so, we start by defining an *algebraic set*, $V(S)$, of a given finitely generated set $S \subset \mathbf{K}[x_1, \ldots, x_n]$, as $V(S) = \{x/\forall f \in S, f(x) = 0\}$. In the special case where $V(S)$ cannot be written as a union of two proper algebraic subsets it is called an *algebraic variety* [44].

Algebraic varieties are of major importance as they provide a one-to-one correspondence between the prime ideals of a ring R and the irreducible closed elements of $Spec(R)$, endowed with Zariski topology, by associating $V(\mathfrak{p})$ to a prime ideal $\mathfrak{p}$.

When the algebraic set is generated by homogeneous polynomials, the variety is said to be *projective* and *affine* otherwise. The *projective closure* W of an affine variety V is defined by the homogenisation of each polynomial generating the corresponding algebraic set. $W - V$ defines a set of points called *points at the infinity* [127]. From a practical point of view, the points at infinity are obtained by setting the homogenisation variable to 0 and solving the resulting system.

The counterpart of the Krull dimension of an algebraic variety V, simply called

dimension of V, is given by the smallest m such that $\mathbf{K}(V)$ is an *algebraic extension* of $\mathbf{K}[X_1 \dots X_m]$. If $I(V)$ denotes the prime ideal generated by the polynomials vanishing on V, then $\mathbf{K}[X_1 \dots X_n]/I(V)$ is an integral domain of Krull dimension equal to the dimension of the algebraic variety V. Hence, given an algebraic varieties V, $I(V)$ is noetherian and the Krull-Akizuki theorem (2.1.4) can be applied to the special case of algebraic varieties of dimension 1. Such objects, called *curves*, give a geometrical representation of one-dimensional noetherian domain.

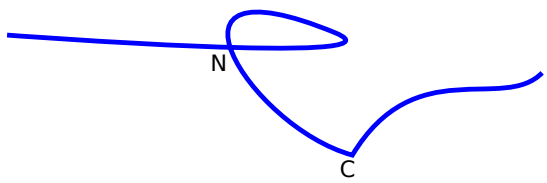

Figure 2.2: One-dimensional noetherian integral domain.

A curve can be either *singular*, if there exists a point where all the partial derivative vanish, or *smooth*, if no such point exists. Figure 2.2 gives a geometrical representation of a one-dimensional noetherian domain, with two singular points, a *node* N and a *cusp* C. Following the Krull-Akizuki theorem (2.1.4), those singularities on the curve can be resolved by taking the integral closure of the corresponding ring. Such a process is called *normalisation*.

In order to have a closer examination of curves, we need to consider further refinements to the theory already presented. As we saw previously, in the general case of R being a one-dimensional noetherian domain over a field $\mathbf{K}$, the prime ideal decomposition is not unique, implying that the *fractional ideals* do not form a group any more. However, restricting our attention to the *invertible ideals* of R, i.e. the fractional ideals $\mathfrak{a}$ for which there exists $\mathfrak{b}$ such that $\mathfrak{a}\mathfrak{b} = R$, leads to a new group structure. And then, the quotient group of the invertible ideals of R by its fractional principal ideals

can be defined. Such group is called the *Picard group* of the ring R and is denoted $Pic(R)$ [100, chapter 1, definition 12.5]. The side effect of this approach is that the information conveyed by the non-invertible ideals is lost. Another solution is then to reintroduce the group law using more abstract and artificial objects, called divisors. The general idea used here, is to construct a group law from a simple collection of objects having no special structure on them.

This formal group, called the *divisor group* of R, is defined as the direct sum of $\mathbb{Z}\mathfrak{p}$ for all the prime ideals $\mathfrak{p}$ of R. A *divisor* is an element of the divisor group of R, noted $\mathrm{Div}(R)$. For $f \in \mathrm{Div}(R)$ it is written as a formal sum $\mathrm{div}(f) = \sum_{\mathfrak{p}} n_{\mathfrak{p}}\mathfrak{p}$. The *degree* of this divisor is given by $\mathrm{Deg}(\mathrm{div}(f)) = \sum_{\mathfrak{p}} n_{\mathfrak{p}}$. In order to mirror the construction of the Picard group, we define a *principal divisor*, for each element of $\mathbf{K}$, as the counterpart of the principal ideal in the former case. Then, quotienting the divisor group of R by its subgroup of principal divisors, leads to a new group called *divisor class group* [100, chapter 1, definition 12.13].

In fact, when R is a Dedekind domain over $\mathbf{K}$, the Picard group and the divisor class group are isomorphic and as such, can be identified. It is also interesting to note that, in this case, they are equal to the *ideal class group* [100, chapter 1, proposition 12.14], which gives a measure of how far R is from being a principal domain. The cardinality of the group is called the *class number* of $\mathbf{K}$.

The divisor theory can be transposed to curves by taking the spectrum of R endowed with Zariski topology. As explained above, if R is a Dedekind domain, then the corresponding curve C is smooth. Looking at the quotient of the divisors of degree 0 by the principal divisors, defines a subgroup of the Picard group, noted $Pic^0(C)$ and called the *Jacobian* of C. From a topological point of view this group is a *complete*, *connected* group variety, i.e. an *abelian variety* [95, chapter 1]. The dimension of the Jacobian of C, as a variety, is given by the *genus* of the curve C.

Theorem 2.2.1 (Riemann-Roch [127, chapter II, §5]). Let C be a smooth curve and c a canonical divisor on C [127, chapter II, §4]. There exists $g \in \mathbb{N}$ such that for every divisor D, $l(D) - l(c - D) = \mathrm{Deg}(D) - g + 1$, where $l(d)$ is the dimension of the vector space $L(D) \bigcup \{0\} = \{f \in C(\mathbf{K}) \ / \ \mathrm{div}(f) + D = 0\} \bigcup \{0\}$. g is called the genus of C.

The genus of a curve is an important invariant which permits the classification of curves. In the special case where C is a smooth plane projective curve of degree n, i.e. a projective curve that can be embedded in the projective plane $\mathbb{P}^2$, its genus is given by $g(C) = \dfrac{(n-1)(n-2)}{2}$. In particular it means that projective plane curves of certain degree do not exist, for instance no such curve can have degree 2. The classification can be improved by defining *isogeny classes*. Two abelian varieties are said to be *isogenous* if there is a surjective morphism with finite kernel, called an *isogeny*, between them. Any abelian variety which is not isogenous to a product of lower-dimensional abelian varieties is said to be *simple*, and in the case where it is isogenous to the power of a simple abelian variety it is called *ordinary*.

By applying the above equality in the case of C being a smooth projective and plane curve of genus 0, we see that C has either degree 1, i.e. C is a copy of $\mathbb{P}^1$, or degree 2, i.e. C is a conic in $\mathbb{P}^2$.

As curves of genus 1 are of more interest here, let C be a smooth projective curve of genus 1 with a rational point at infinity $\mathcal{O}$. For $D \in \mathrm{Div}(C)$ and $f \in L(D)$, $0 = \mathrm{Deg}(\mathrm{div}(f)) \geq \mathrm{Deg}(-D) = -\mathrm{Deg}(D)$, which implies $\mathrm{Deg}(D) \geq 0$.

Applying the Riemann-Roch theorem (2.2.1) to $D = c$, yields $\mathrm{Deg}(c) = 2g - 2$. Thus, $L(c - D) = \emptyset$ and $l(c - D) = 0$. But as C is a genus 1 curve $l(D) = \mathrm{Deg}(D)$.

Considering, $D = D_n = n(\mathcal{O})$, and $L_n = L(n(\mathcal{O}))$, for $n \in \mathbb{N}$, yields

$$\begin{aligned}
L_1 &= \mathbf{K}, \\
L_2 &= \mathbf{K} \oplus \mathbf{K}x, \\
L_3 &= \mathbf{K} \oplus \mathbf{K}x \oplus \mathbf{K}y, \\
L_4 &= \mathbf{K} \oplus \mathbf{K}x \oplus \mathbf{K}y \oplus \mathbf{K}x^2, \\
L_5 &= \mathbf{K} \oplus \mathbf{K}x \oplus \mathbf{K}y \oplus \mathbf{K}x^2 \oplus \mathbf{K}xy, \\
L_6 &\supsetneq \mathrm{Vect}_{\mathbf{K}}\{1, x, x^2, x^3, y, xy, y^2\}
\end{aligned}$$

where x and y are two functions with only one pole of order 2 in $\mathcal{O}$ and one pole of order 3 in $\mathcal{O}$ respectively. As L_6 is of dimension 6 but contains seven elements, it leads to the following equation, for some $a_i \in \mathbf{K}$:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.2.1}$$

Hence, genus one curves can be defined by equation 2.2.1, named the *Weierstrass equation*. In fact, the curves it defines are not necessarily smooth, however the extra condition on the *discriminant* Δ of the equation not being zero leads to a proper definition of the class of smooth projective curve of genus 1, whose objects are called *elliptic curves* (figure 2.3).

A case of interest for elliptic curves is when an isogeny class is defined by an isomorphism over an algebraic closure $\overline{\mathbf{K}}$. In fact, if a curve $E/\mathbf{K}$ and a curve $E'/\mathbf{K}$ are isomorphic over $\overline{\mathbf{K}}$, then E' is said to be a *twist* of E. For a curve E defined by equation

2.2.1, the following variables can be defined [127, chapter III, §1]:

$$
\begin{aligned}
b_2 &= a_1^2 - 4a_2 & c_4 &= b_2^2 - 24b_4 \\
b_4 &= a_1a_3 + 2a_4 & c_6 &= -b_2^3 + 36b_2b_4 - 216b_6 \\
b_6 &= a_3^2 + 4a_6 & \Delta &= -b_2^2b_8 + 9b_2b_4b_6 - 8b_4^3 - 27b_6^2 \\
b_8 &= a_1^2a_6 - a_1a_3a_4 + a_2a_3^2 + 4a_2a_6 - a_4^2 & j &= \tfrac{c_4^3}{\Delta}
\end{aligned}
$$

While $\Delta \neq 0$, the discriminant of the curve, depends on each curve, j is a constant for all the curves inside an isomorphism class, and as such is called the *j-invariant*.

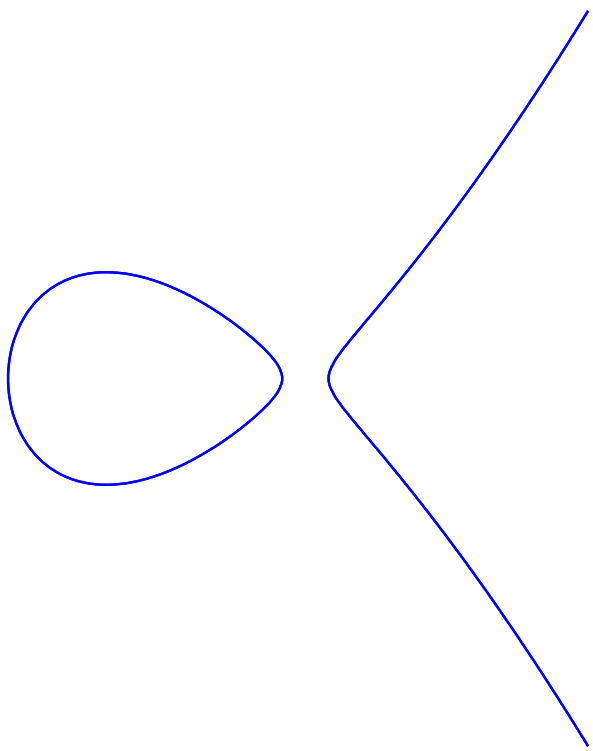

Figure 2.3: Elliptic curve of equation $y^2 = x^3 - 2x + 1$.

One of the most important properties of smooth genus 1 curves is that they are the

only curves which are isomorphic to their Jacobian, and as such they inherit its group law. To describe it geometrically we start by remarking that homogenising equation 2.2.1 leads to only one point at infinity $\mathcal{O} = [0 : 1 : 0]$. Then, we state the following important result.

Theorem 2.2.2 (Bezout[44, chapter 5, section 5.3]). Let C_1 and C_2 be two plane projective curves over an algebraically closed field $\mathbf{K}$, such that they are defined by different irreducible polynomials. The total number of intersection points of C_1 and C_2, counted with their multiplicity, is given by the product of their degrees.

As a consequence the number of intersection points of an elliptic curve and a line is three, allowing us to properly define the addition law. Let P and Q be two points on E, then the line connecting them will intersect E in a third point R. The same idea can be used to define R', the third intersection point of E and the line connection R and $\mathcal{O}$. R' is defined as being $P + Q$ (figure 2.4).

Considering the Weirstrass equation 2.2.1 for an elliptic curve E over a field $\mathbf{K}$, it is possible to state an analytic version of the sum of two points, $P = (x_P, y_P)$ and $Q = (x_Q, y_Q)$, by studying the points of intersection of the two lines with the curve. The first thing to remark is that the line connecting $\mathcal{O}$ and P intersects E in the point $-P = (x_P, -(y_P + a_1 x_P + a_3))$. This means that, the line connecting P and Q will touch E in $R = -(P+Q)$ and then applying the formula will lead to $R' = P+Q$. Two cases may arise depending whether or not P has intersection multiplicity 2. If $P \neq Q$, then define

$$\lambda = \frac{y_Q - y_P}{x_Q - x_P}, \quad \mu = \frac{y_P x_Q - y_Q x_P}{x_Q - x_P},$$

and if $P = Q$, let

$$\lambda = \frac{3x_P^2 + 2a_2 x_P + a_4 - a_1 y_P}{2y_P + a_1 x_P + a_3}, \quad \mu = \frac{-x_P^3 + a_4 x_P + 2a_6 - a_3 y_P}{2y_P + a_1 x_P + a_3}.$$

Thus the line intersecting E in P and Q has equation $y = \lambda x + \mu$, which yields

$$P + Q = (\lambda^2 + a_1\lambda - a_2 - x_P - x_Q, -(\lambda + a_1)x_{P+Q} - \mu - a_3).$$

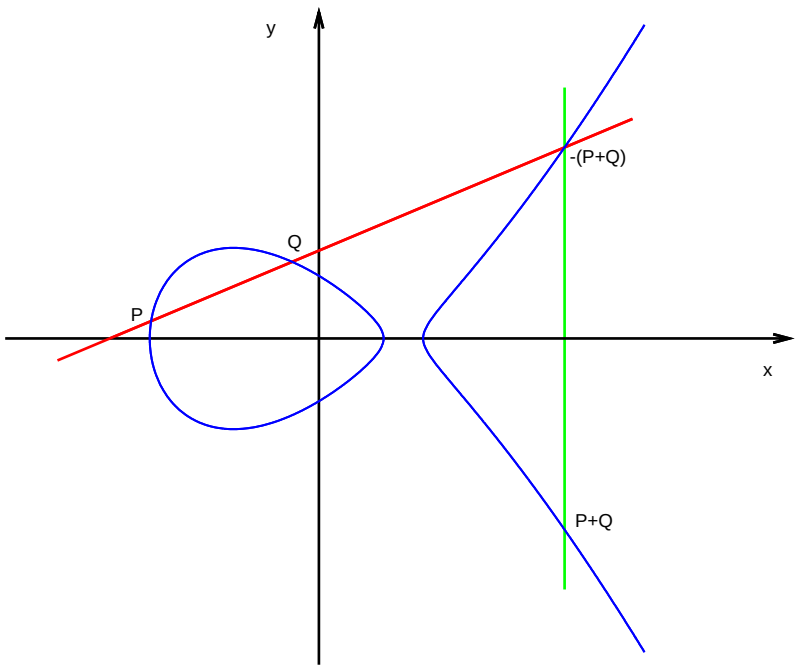

Figure 2.4: Geometrical representation of elliptic curve addition law.

Although, we defined elliptic curves from an abstract point of view as abelian varieties of dimension 1, it is possible to link this to their historical definition. Everything started during the 18th century when the question of the arc length of an ellipse was raised. This led to the study of integrals involving the square root of polynomials of degree 3 or 4. It was quickly discovered that such integrals cannot be expressed using familiar functions, which were named *elliptic integrals*.

Abel, then, had the idea of studying the inverse of these elliptic integrals , which are now known as elliptic functions. It turned out that such functions are doubly periodic, i.e. given an elliptic function $f(x)$ $\exists$ $\omega_1, \omega_2 \in \mathbb{C}$ such that $\frac{\omega_1}{\omega_2} \in \mathbb{R}$ and

$f(x+\omega_1) = f(x+\omega_2) = f(x)$. This means that $f(x)$ is isomorphic to the lattice $\Lambda \subset \mathbb{C}$ generated by ω_1, ω_2. Then, considering C, the curve defined by f, and $\Omega(C)$, the space of functions which are differentiable at every point in C, $\mathrm{Hom}(\Omega(C), C)/\Lambda$ is, in fact, the Jacobian of C [60].

Although higher genus curves are of great interest for number theory and solving *diophantine equations*, we will only define an *abelian surface* as being a genus 2 abelian variety, and instead will focus on maps from the Jacobian of a curve to its base field.

2.3 Pairings

Let J be the Jacobian variety of dimension g of a curve C over a field $\overline{\mathbf{K}}$. The group of $\mathbf{K}$-rational points of J is denoted by $J(\mathbf{K})$, and then the *r-torsion* subgroup of $J(\overline{\mathbf{K}})$, $J(\overline{\mathbf{K}})[r]$, containing the elements of order r, for r coprime to the characteristic of $\mathbf{K}$, is isomorphic to $(\mathbb{Z}/r\mathbb{Z})^{2g}$ [68]. We define μ_r, as its counterpart over $\overline{\mathbf{K}}$, i.e. the set of the rth-roots of unity. Given a divisor D on $J(\overline{\mathbf{K}})$, $r_J^* D$ is linearly equivalent to rD, with r_J the following isogeny over $J(\overline{\mathbf{K}})$: $x \mapsto rx$, and r_J^* its *dual* [95, chapter I, section 8]. As rD is in $J(\overline{\mathbf{K}})$ it is a degree zero divisor and as such there exists two functions $(f_1) = rD$ and $(f_2) = r_J^* D$. The object of interest here is $\mathrm{div}(f_1 \circ r_J)$.

$$\mathrm{div}(f_1 \circ r_J) = r_J^*(\mathrm{div}(f_1)) = r_J^*(rD) = r(r_J^* D) = r(\mathrm{div}(f_2)) = \mathrm{div}(f_2^r)$$

This equality means that $\dfrac{f_2^r}{f_1 \circ r_J} = c$ for c some constant function. Then for $P \in J(\overline{\mathbf{K}})[r]$ and $x \in J(\overline{\mathbf{K}})$

$$f_2(x+P)^r = c.f_1 \circ r_J(x+P) = c.f_1(rx+rP) = c.f_1(rx) = f_2(x)^r$$

Therefore, $\left(\dfrac{f_2(x)}{f_2(x+P)}\right)^r = 1$. Hence, for $Q \in J(\overline{\mathbf{K}})[r]$ having divisor D we can

define the following map:

$$\begin{array}{rcccl} e_r : J(\overline{\mathbf{K}})[r] & \times & J(\overline{\mathbf{K}})[r] & \to & \mu_r \\ (P & , & Q) & \mapsto & \frac{f_2(Q)}{f_2(Q+P)} \end{array}$$

As expected, the map e_r, called the *Weil pairing*, maps two points on the Jacobian of C to a point on its base field. The Weil pairing defined from two group $\mathbb{G}_1$ and $\mathbb{G}_2$ into a group $\mathbb{G}_T$ has the following important properties:

Proposition 2.3.1 ([95, chapter 1, section 13]). *The Weil pairing is*

- *non-degenerated:*
  - $\forall P \in \mathbb{G}_1, P \neq 0 \quad \exists Q \in \mathbb{G}_2$ *such that* $e_r(P,Q) \neq 1$
  - $\forall Q \in \mathbb{G}_2, Q \neq 0 \quad \exists P \in \mathbb{G}_1$ *such that* $e_r(P,Q) \neq 1$
- *bilinear:* $\forall P, P' \in \mathbb{G}_1$ *and* $\forall Q, Q' \in \mathbb{G}_2$

$$e(P+P',Q) = e(P,Q)e(P',Q) \text{ and } e(P,Q+Q') = e(P,Q)e(P,Q')$$

- *Galois-invariant:* $\forall \sigma \in \mathrm{Gal}(\overline{\mathbf{K}}/\mathbf{K}) \quad e_r(\sigma(P),\sigma(Q)) = \sigma(e_r(P,Q))$

Given an isogeny ϕ, $e_r(\phi(a_1), a_2) = e_r(a_1, \phi^*(a_2))$.

An other important pairing is the *Tate pairing*. In the scope of an elliptic curve E, it is defined as follows:

$$\begin{array}{rcccl} e_r : E(\mathbf{K})[r] & \times & E(\mathbf{K})/rE(\mathbf{K}) & \to & \mathbf{K}^*/(\mathbf{K}^*)^r \\ (P & , & Q) & \mapsto & f(D) \end{array}$$

where f is a function whose divisor is equivalent to $r(P) - r(\mathcal{O})$, and D is a degree 0 divisor equivalent to $(Q) - (\mathcal{O})$ and has support disjoint from that of f.

The pairing e_r maps a pair of points on the elliptic curve to an equivalence class of $\mathbf{K}^*/(\mathbf{K}^*)^r$. However by pointing out the existence of an isomorphism between the elements of order r in $\mathbf{K}$ and $\mathbf{K}^*/(\mathbf{K}^*)^r$, it becomes possible to compute $f(D)$ as $f(Q)$. In the simple case where $\mathbf{K}$ is algebraically closed μ_r is a subset of $\mathbf{K}$ implying that $\mathbf{K}^*/(\mathbf{K}^*)^r$ is isomorphic to μ_r. More generally, $\mathbf{K}^*/(\mathbf{K}^*)^r$ is isomorphic to μ_d where d divides r.

The Tate pairing satisfies the following properties.

Proposition 2.3.2 ([46]). *The Tate pairing is bilinear, non-degenerated and Galois invariant.*

Throughout the next chapter, we will take a closer look at pairings over finite fields, answering the question as to which curves are of interest in this context and how to compute a pairing in practice.

3

Pairings

The world let by its own follows inevitable laws.

H. Bergson

3.1 Pairings over finite fields

The definition of pairings given in the preceding chapter is valid for any abelian variety J over any field $\mathbf{K}$. However, in order for the pairing to exist, if one wants to restrict oneself to a finite field $\mathbf{K}$, the first thing to do is to ensure that $\mathbf{K}$ contains the rth roots of unity. Given a finite field $\mathbb{F}_q$, this is done by adjoining a primitive rth-root of unity ζ_r to $\mathbb{F}_q$ and considering the extension $\mathbb{F}_q(\zeta_r)$ of $\mathbb{F}_q$. The dimension of this extension is called the *embedding degree* of the abelian variety $J/\mathbb{F}_q$. From a practical point of view, the embedding degree of $J/\mathbb{F}_q$ with respect to r, is the smallest integer k such that $r|q^k - 1$. This means that $r \nmid q^i - 1$ for all integers $1 \leq i < k$. This remark links embedding degree and *cyclotomic polynomials*.

Although cyclotomic polynomials and cyclotomic fields are of great interest in number theory we will focus only on a few properties, referring the reader to [138] for more details on the topic. In our case we define the k-th cyclotomic polynomial

as the minimal polynomial of ζ_k, a primitive k-th root of unity in $\overline{\mathbb{Q}}$, and denote it $\Phi_k(X) = \prod_{gcd(k,j)=1} (X - \zeta_k^j)$. For all k the polynomial $\Phi_k(X)$ is defined over $\mathbb{Z}[X]$, and has degree $\varphi(k)$, where φ is *Euler's totient function.* Another formulation of cylotomic polynomials is recursively given as follows:

$$\begin{cases} X^k - 1 &= \prod_{d|k} \Phi_d(X) \\ \Phi_1 &= (X-1) \end{cases}$$

Example 3.1.1. Examples of cyclotomic polynomials are listed below:

$$\begin{aligned} \Phi_2 &= X+1 \\ \Phi_3 &= X^2+X+1 \\ \Phi_6 &= X^2-X+1 \\ \Phi_{12} &= X^4-X^2+1 \end{aligned}$$

For instance when $k = 2$ applying the above recursive definition yields:

$$\begin{aligned} X^2 - 1 &= \Phi_1(X) \cdot \Phi_2(X) \\ \Phi_2 &= \tfrac{(X^2-1)}{(X-1)} \\ &= X+1 \end{aligned}$$

For $k = 6$, $\Phi_6 = \frac{X^6-1}{\Phi_3\Phi_2\Phi_1} = X^2 - X + 1$.

The above remark, together with the following lemma [138, lemma 2.9] helps to view the embedding degree from a new angle.

Lemma 3.1.2. *Let k be an integer, and r a prime not dividing k, then $r|\Phi_k(q)$ is equivalent to saying that k is the multiplicative order of q mod r.*

Since the assertion "k is the smallest integer such that $r|q^k - 1$" is equivalent to "k is the multiplicative order of q mod r", it follows that the definition of the embedding

degree can be stated in terms of cyclotomic polynomials. The result contained in [42, proposition 2.4] is fundamental to the understanding of pairings over finite fields:

Proposition 3.1.3. *Let k be a positive integer, $E/\mathbb{F}_q$ a curve with hr points where r is prime, and let t be the trace of $E/\mathbb{F}_q$. If $r \nmid kq$, then E has embedding degree k with respect to r if and only if $\Phi_k(q) \equiv 0 \mod r$.*

By looking at the definition of the embedding degree, clearly, taking r as a large divisor of $\#J(\mathbb{F}_q) \approx q^g$ [96] implies that most of the elements in $\mathbb{F}_r$ have large order, and so will be the embedding degree k. This is a problem as it renders the computation of the pairings too complex to be used in practice. This gives rise to the notion of *pairing-friendly abelian variety*, i.e. an abelian variety appropriate for being used in the context of pairings. When such varieties have genus 1 they are called *pairing-friendly elliptic curves.*

One way to measure how efficient a pairing-friendly abelian variety is, is to define ρ, the ratio of the size of $J(\mathbb{F}_q)$ to the size of the group of order r, onto which the pairing maps:

$$\rho = g\frac{\log q}{\log r}$$

For a pairing-friendly abelian variety to result in an efficient implementation it is expected to have a ρ-value close to 1.

Unless explicitly mentioned the abelian varieties now considered will be of genus 1, i.e. elliptic curves.

3.2 Pairing-friendly elliptic curves

Let $q = p^m$ for p a prime and m a positive integer. $\mathbb{F}_q$ seen as an m-dimensional vector space over $\mathbb{F}_p$, allows the representation of any endomorphism f of $\mathbb{F}_q$ as a matrix M. The sum of the eigenvalues of M is called the *trace* of the endomorphism

f. When f is the *Frobenius* endomorphism, i.e. $f(x) = \pi_p(x) = x^p, x \in \mathbb{F}_q$, its trace t is linked to the number of $\mathbb{F}_q$-rational points on an elliptic curve $E/\mathbb{F}_q$ by the relation $\#E(\mathbb{F}_q) = q + 1 - t$. As a function t satisfies the *Hasse bound* [127, Theorem V.1.1]: $|t| \leq 2\sqrt{q}$.

The definition given for a pairing-friendly abelian variety being quite vague, we start by defining it more formal in the context of elliptic curves.

Definition 3.2.1. ([43, definition 2.3]) Let E be an elliptic curve defined over $\mathbb{F}_q$. E is said to be *pairing-friendly* if there is a prime $r \geq \sqrt{q}$ dividing $\#E(\mathbb{F}_q)$ and the embedding degree of E with respect to r is less than $\log_2(r)/8$.

With this definition, pairing-friendly elliptic curves are rare [86] and as such hard to find. However, it is still possible to exhibit a few of them using some special constructions based on the so-called *Complex Multiplication (CM) method* [98].

The first elliptic curves to be recognised as pairing-friendly, do not require any specific construction as they are the elliptic curves satisfying $gcd(t, q) > 1$. Such curves, discovered by Menezes, Okamoto and Vanstone [91], are called *supersingular* elliptic curves. Another way to define them is to say that they do not have any p-torsion points.

More generally, any abelian variety of higher dimension that is isogenous to a product of supersingular elliptic curves is called a *supersingular abelian variety.* In the case of supersingular elliptic curves it was shown that they can have five possible embedding degrees k, corresponding to five possible absolute values of the trace of Frobenius t (table 3.1) [91].

Since using only supersingular curves is a bit restrictive, new methods have been developed to construct *ordinary* pairing-friendly, elliptic curves. One of the first suggested was by Cocks and Pinch [46]. Using their method, it is easy to generate pairing-friendly elliptic curves of arbitrary embedding degree. However a major drawback is their ρ-value, close to 2, leads to "slow" implementations.

k	t	$\#E(\mathbb{F}_q)$	p, m
1	$\pm 2\sqrt{q}$	$q \mp 2\sqrt{q} + 1$	any p, m even
2	0	$q + 1$	any p, any m
3	$\pm\sqrt{q}$	$q \mp \sqrt{q} + 1$	$p \equiv 2 \bmod 3$, m even
4	$\pm\sqrt{2q}$	$q \mp \sqrt{2q} + 1$	$p = 2$, m odd
6	$\pm\sqrt{3q}$	$q \mp \sqrt{3q} + 1$	$p = 3$, m odd

Table 3.1: Classification of supersingular elliptic curves.

Leaving those curves aside, few *families* of pairing-friendly elliptic curves with ρ-value close to 1 remain. They are constructed using the CM Method [15]. The idea relies on the possibility to factor $4p - t^2$ into a square v^2 multiplied by a small number D, called the *CM discriminant*. Using the CM method with the parameters p and $n = p+1-t$ allows the constructions of curves with n points over $\mathbb{F}_p$. In fact, all known non-supersingular pairing-friendly elliptic curves are CM curves, which means that they satisfy the CM equation $Dv^2 = 4p - t^2$ for some small discriminant $D < 10^{14}$ [130]. Its is interesting to note that the only reason to keep the discriminant D small is to render the construction possible.

MNT curves: Miyaji, Nakabayashi and Takano [97], introduced the first ordinary pairing-friendly elliptic curves. Their strategy consist in solving a generalised Pell equation of the form $X^2 - SDV^2 = M$ for some small discriminants until the solution provides suitable parameters to be used by the CM method. The major downside of this construction relies on the fact that, given a discriminant D, it leads to very few curves, even without any bound on the field size. Although they are rare, it happens that they can be useful in practice and have ρ-value close to 1. Table 3.2 gives a characterisation of a family of ordinary elliptic curves with embedding degree $k = 3, 4$ or 6, known as the MNT curves.

When $k = 6$, i.e. $p = 4x^2 + 1$ and $t = 1 \pm 2x$ (table 3.2), and X is set to $6x \pm 1$ the CM equation can be rewritten as the generalised Pell equation $X^2 - 3Dv^2 = -8$. Then solving it, leads to the MNT family of elliptic curves with embedding degree 6 which

k	$q = p$	t
3	$12x^2 - 1$	$-1 \pm 6x$
4	$x^2 + x + 1$	$-x, x + 1$
6	$4x^2 + 1$	$1 \pm 2x$

Table 3.2: Classification of MNT curves ($x \in \mathbb{Z}$).

can be described by the following three polynomials, $t(x)$, $r(x)$ and $p(x)$ representing the prime modulus p, the group order r and the trace t, respectively:

$$\begin{aligned} t(x) &= x + 1 \\ r(x) &= x^2 - x + 1 \\ p(x) &= x^2 + 1. \end{aligned}$$

Freeman curves: Similarly to MNT curves, Freeman curves [41] form a sparse family. Since their discriminant must satisfy $D \equiv 43$ or $67 \mod 120$, it is usually very large. If there exists D such that the Pell equation $X - 15Dy^2 = -20$ has a solution, then this yields a family of curves, having ρ-value 1 and embedding degree 10, defined by the following polynomials:

$$\begin{aligned} t(x) &= 10x^2 + 5x + 3 \\ r(x) &= 25x^4 + 25x^3 + 15x^2 + 5x + 1 \\ p(x) &= 25x^4 + 25x^3 + 25x^2 + 10x + 3. \end{aligned}$$

BN curves: Barreto and Naehrig curves [9] are probably the most well known pairing-friendly elliptic curves. While they also have ρ-value 1, BN curves, on the contrary to the previously mentioned curves, are plentiful, easy to find and have a constant discriminant $D = 3$. This family of curves having embedding degree $k = 12$

can be defined using the following polynomials:

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

BW curves: Brezing and Weng [19], gave some more general constructions of pairing-friendly elliptic curves, making heavy use of cyclotomic fields. Although their construction produces curves having ρ-value slightly larger than 1, they cover almost all possible embedding degrees. The two following families feature a discriminant of $D = 3$.

For $k = 8$, $\rho = 5/4$, the family is defined as follows:

$$\begin{aligned} t(x) &= x^5 - x + 1 \\ r(x) &= x^8 - x^4 + 1 \\ p(x) &= \frac{1}{3}(x^{10} - 2x^9 + x^8 - x^6 + 3x^5 - x^4 + x^2 - 2x + 1). \end{aligned}$$

For $k = 32$, $\rho = 17/16$, and the curves are defined by the polynomials:

$$\begin{aligned} t(x) &= x^{17} - x + 1 \\ r(x) &= x^{32} - x^{16} + 1 \\ p(x) &= \frac{1}{3}(x^{34} - x^{33} + x^{32} - x^{18} + 2x^{17} - x^{16} + x^2 - 2x + 1). \end{aligned}$$

KSS curves: The KSS construction [76] is quite similar to the one used by Brezing and Weng as it only differs on how to define the chosen cyclotomic field. This new approach allowed improvements on some ρ-values of BW curves. Embedding degrees $k = 8$, 16, 32, 36 and 40 can be achieved, with a special mention for curves of embedding degree 18 having ρ-value 4/3. In the case of embedding degree $k = 8$, $\rho = 3/2$ and

$D = 1$,

$$\begin{aligned} t(x) &= \frac{1}{15}(2x^3 - 11x + 15) \\ r(x) &= \frac{1}{450}(x^4 - 8x^2 + 25) \\ p(x) &= \frac{1}{180}(x^6 + 2x^5 - 3x^4 + 8x^3 - 15x^2 - 82x + 125). \end{aligned}$$

For embedding degree $k = 18$, $\rho = 4/3$, and $D = 3$,

$$\begin{aligned} t(x) &= \frac{1}{7}(x^4 + 16x + 7) \\ r(x) &= \frac{1}{343}(x^6 + 37x^3 + 343) \\ p(x) &= \frac{1}{21}(x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401). \end{aligned}$$

In this case we note that $t(x)$, $r(x)$ and $p(x)$ must evaluate as integers and so $x \equiv 14 \mod 42$ [76].

Hence, two basic choices of pairing-friendly elliptic curves are available, the supersingular curves over any finite field of characteristic 2 or 3, and ordinary pairing-friendly elliptic curves over $\mathbb{F}_p$. In the former case only curves with embedding degrees up to $k = 6$, over fields of characteristic 3, are possible. Fortunately, ordinary pairing-friendly elliptic curves also exist, allowing an unlimited choice of k. Given that, the construction of pairing-friendly elliptic curves with any embedding degree becomes possible, meaning long term viability of systems using them, as long as these curves remain efficient enough.

It is also interesting to note that all elliptic curves over finite fields have a quadratic twist, i.e. are isomorphic to a curve over $\mathbb{F}_{q^2}$. However, under certain conditions [43, section 7.3] it may happen that curves have higher order twist equal to $3, 4$ or 6. In fact only curves with CM discriminant $D = 1$, i.e. curves of the form $y^2 = x^3 + ax$, have quartic twist, while the ones having a CM discriminant $D = 3$, that is the curves

of equation $y^2 = x^3 + b$, can have cubic and sextic twists. Although the case of curves over characteristic 2 or 3 is more complicated they still have the degree of their twists dividing 6 [127].

In practice, when implementing pairings on ordinary pairing friendly curves, a parameter is taken on the curve defined over the base field $\mathbb{F}_p$, while the other one is picked on a twisted curve featuring a group of points of order r which are isomorphic to a group of points on the curve defined over $\mathbb{F}_{p^k}$. This is very useful as the output of the Tate pairing can then be taken as an element of $\mathbb{F}_{q^{k/d}}$ if the degree d of the twist divides the embedding degree k. This idea, known as "compression technique" as $\lceil \log_2 d \rceil$ bits of information are dropped, was first introduced by Scott and Barreto [122] in the case of a quadratic twist before being extended to sextic twists [9].

Another important efficiency improvement in the computation of a pairing resulting from the use of the twist is that the second parameter can be taken in $E'(\mathbb{F}_{p^{k/d}})$ instead of $E(\mathbb{F}_{p^k})$. This clearly allows more efficient implementations. For example noting that BN curves have a CM discriminant $D = 3$ and an embedding degree $k = 12$ implies that they admit a sextic twist. Hence, the second parameter can be picked in $E'(\mathbb{F}_{p^2})$ instead of $E(\mathbb{F}_{p^{12}})$, implying a lower computational cost.

For a more complete study of pairing-friendly elliptic curves the reader should refer to Freeman, Scott and Teske taxonomy [43].

3.3 Computing pairings

Once the definition of a pairing has been given and some pairing-friendly elliptic curves have been presented, the next stage clearly is to explain how to compute a pairing in practice. Although the Weil and Tate pairings were introduced we will focus only on the latter, the former being usually less efficiently implementable.

From the definition of the Tate pairing given at the end of chapter 2, it appears

that the main question is how to find a function f having a divisor equivalent to $r(P) - r(\mathcal{O})$. In fact, Miller's idea [93] to compute the Weil pairing, can be adapted to the computation of the Tate pairing. The key point is the evaluation of $f(R)$ for every R in the support of D. Following Miller one can randomly pick R on the curve and for all $i \leq r$ define f_i such that $\mathrm{div}(f_i) = i(P+R) - i(R) - ([i]P) + (\mathcal{O})$. By construction, when $i = r$, $rP = \mathcal{O}$, and $f_r = f$.

For any 2 points A and B, denote by $l_{A,B}$ the line passing through A and B, and v_A the vertical line connecting A to $\mathcal{O}$. Remarking the following two equalities,

$$\mathrm{div}(l_{i_1P,i_2P}) = ([i_1]P) + ([i_2]P) + (-[i_1+i_2]P) - 3(\mathcal{O})$$

$$\mathrm{div}(v_{[i_1+i_2]P}) = ([i_1+i_2]P) + (-[i_1+i_2]P) - 2(\mathcal{O})$$

leads to

$$\mathrm{div}(f_{i_1+i_2}) = \mathrm{div}(f_{i_1}) + \mathrm{div}(f_{i_2}) + \mathrm{div}(l_{i_1P,i_2P}) - \mathrm{div}(v_{[i_1+i_2]P}).$$

Hence,

$$\begin{cases} f_{i_1+i_2} & = & \frac{f_{i_1} f_{i_2} l_{[i_1]P,[i_2]P}}{v_{[i_1+i_2]P}}, \\ f_0 & = & 1, \\ f_1 & = & \frac{l_{P,R}}{v_{P+R}} \end{cases}$$

defines a recursive sequence of length $r+1$, whose last element is f. This allows us to derive Miller's algorithm for Tate pairing as described in algorithm 3.1.

As explained above, the Miller loop computes the function f. However an extra stage is required in order to ensure the uniqueness of the result. Since the output of the loop does not necessarily have order r, an extra exponentiation of f to the power $(q^k - 1)/r$ must be performed. This results in $\log(r)$ squaring, $H(r) - 1$ multiplications, with $H(r)$ the Hamming weight of r, i.e. the number of 1's in the binary representation

Algorithm 3.1 Miller's algorithm for Tate pairing.

INPUT: $P \in E(\mathbb{F}_q)[r], Q \in E(\mathbb{F}_q)$.
OUTPUT: $e(P, Q)$.

```
T ← P
f ← 1
for i ← ⌊log(r) − 1⌋ to 0 do
    f ← f^2 l_{T,T}(Q) / v_{2T}(Q)
    T ← 2T
    if r_i = 1 then
        f ← f l_{T,P}(Q) / v_{T+P}(Q)
        T ← T + P
    end if
end for
f ← f^{(q^k−1)/r}
return f
```

of r, and an exponentiation. As such, Miller's algorithm is a polynomial time algorithm.

The version of Miller's algorithm given here (algorithm 3.1) is basic and some more sophisticated ones can be derived, [120, 66, 32]. Moreover, when dealing with an implementation of pairings, one should keep in mind Galbraith's list of seven common strategies that can be used in order to improve the efficiency of pairing computations [46, chapter IX].

1. Pick an r having small hamming weight.

2. As much as possible work in $\mathbb{F}_q$ instead of $\mathbb{F}_{q^k}$.

3. Use efficient arithmetic over $\mathbb{F}_q$ and $\mathbb{F}_{q^k}$.

4. Avoid expensive calculation like division, use more squaring and less multiplications.

5. Find ways to improve the efficiency of the final exponentiation.

6. Note that whenever $k > 1$ and n is prime, n does not divide $q - 1$ implying that the $(q^k - 1)/n$th power of all elements of $\mathbb{F}_q^*$ is 1. Therefore all terms in the

algorithm leading to an element of $\mathbb{F}_q^*$ can be ignored.

7. If $P \in E(\mathbb{F}_q)[r]$ and $Q = (X, Y) \in E(\mathbb{F}_{q^k})$ with $X \in \mathbb{F}_{q^{k/2}}$, then the denominators computation in the Miller loop can be discarded [10].

It is interesting to note that faster variants of the Tate pairing, like ate or R-ate pairings, exist. Although we do not consider their specifics, as it is not the main topic of interest here, the reader can refer to [39, 82], for more details on the subject. Instead, we will present new results on the field into which the pairing maps.

4

Pairings and the minimal embedding field

I revolt, therefore we are.

A. Camus

Following the definition of pairings (chapter 2) and its restriction to the case of a finite field $\mathbb{F}_q$ (chapter 3), we see that the set of the rth roots of unity $\mu_r \subset \overline{\mathbb{F}}_q$ must be contained in an extension field $\mathbb{F}_{q^k}$, where k is the embedding degree of the pairing-friendly abelian variety used. In fact, Rubin and Silverberg [112] and Hitt [69] observed that when the field size q is not prime, the rth roots of unity may be contained in a proper subfield $F \subset \mathbb{F}_{q^k}$. This observation leads to the definition of the *minimal embedding field* of a pairing-friendly abelian variety J over $\mathbb{F}_q$, with respect to r, as the smallest field $F \subset \mathbb{F}_{q^k}$ containing μ_r. An obvious question is then to know when the minimal embedding field is not a proper subfield of $\mathbb{F}_{q^k}$.

Rubin and Silverberg [113] have given an answer to this question in the case where J is supersingular by demonstrating a lower bound on r that guarantees that the minimal embedding field is $\mathbb{F}_{q^k}$. Their bound depends on q and on the dimension g of

the supersingular abelian variety, but does not depend on k. So, in order to generalise their result we want to give explicit conditions on q, r, and k that guarantees that the minimal embedding field of an abelian variety $J/\mathbb{F}_q$, supersingular or not, is in fact $\mathbb{F}_{q^k}$.

4.1 Framework

For J an abelian variety over $\mathbb{F}_q$ with embedding degree k, we know that $\mathbb{F}_{q^k}$ is the smallest extension of $\mathbb{F}_q$ containing the rth roots of unity. Then, the Weil pairing [127, §III.8] and [96, §16] and the Tate pairing [35] take values in a subgroup and a quotient group of $\mathbb{F}_{q^k}^*$, respectively. The key observation made by Rubin and Silverberg [112] and Hitt [69] is that these pairings actually take values in the minimal embedding field and that this field may be a proper subfield of $\mathbb{F}_{q^k}$. This observation, found in different forms in each paper, is expressed by Hitt as follows:

Lemma 4.1.1 ([69, Lemma 1]). *Let $q = p^m$ for some prime p and positive integer m, let $r \neq p$ be a prime, and let k be the smallest integer such that r divides $q^k - 1$. Then*

$$k = \frac{\mathrm{ord}_r(p)}{\gcd(\mathrm{ord}_r(p), m)},$$

where $\mathrm{ord}_r(p)$ is the order of p in $(\mathbb{Z}/r\mathbb{Z})^*$.

The main consequence of the result is that the minimal embedding field of an abelian variety $J/\mathbb{F}_q$ is $\mathbb{F}_{q^{k'}}$, where $k' = \mathrm{ord}_r(p)/m \in \mathbb{Q}$, and as such, is not necessarily $\mathbb{F}_{q^k}$ (figure 4.1).

Indeed, Hitt gives examples of abelian varieties where $k/k' = m$, which is the largest possible ratio for these parameters [69, §4]. One important thing to note is that when the abelian variety is defined over a prime field, i.e. $m = 1$, Hitt's lemma has no effect, as the minimal embedding field is always $\mathbb{F}_{q^k} = \mathbb{F}_{p^k}$. Thus, only the case of pairing-friendly abelian varieties over extension fields needs to be considered.

$$\begin{tikzcd}
 & \mathbb{F}_{q^k} \arrow[d, dash, "\geq 1"] \\
\mathbb{F}_q \arrow[ur, dash, "k"] \arrow[d, dash, "m"'] & \mathbb{F}_{q^{k'}} \\
\mathbb{F}_p \arrow[ur, dash, "\mathrm{ord}_r(p)"'] &
\end{tikzcd}$$

Figure 4.1: Field diagram showing the minimal embedding field $\mathbb{F}_{q^{k'}}$.

In the case where $J/\mathbb{F}_q$ is supersingular and elementary, with embedding degree k, and $r \nmid 2k$, Rubin and Silverberg defined the exponent c_J as the smallest half-integer such that r divides $q^{c_J} - 1$. Then, their theorem, phrased in terms of c_J, gives conditions on q, r, and k for the minimal embedding field to be $\mathbb{F}_{q^k}$.

Theorem 4.1.2 ([112, Theorem 7] and [113, Theorem 6.3]). Suppose J is an elementary supersingular abelian variety of dimension g over $\mathbb{F}_q$, $q = p^m$, $r \neq p$ is a prime divisor of $\#J(\mathbb{F}_q)$, and s is the multiplicative order of p mod r. Let $F_J(x) \in \mathbb{Z}[x]$ be the characteristic polynomial of the Frobenius for J, and let f be the unique integer such that $F_J(x)^{1/f}$ is irreducible in $\mathbb{Z}[x]$. If q is a square, assume $r > (1+p)^{mg/2f}$. If q is not a square, assume $r > (1+\sqrt{p})^{2mg/3f}$ and $r > 7$. Then $p^s = q^{c_{J,q}}$, so $\mathbb{F}_{q^{c_{J,q}}}$ is the smallest extension of $\mathbb{F}_p$ whose multiplicative group has a subgroup of order r.

In order to improve the bounds, and extend their validity to all abelian varieties, we start by linking the minimal embedding field to cyclotomic polynomials, in the same way as done to relate embedding degree and cyclotomic polynomials.

Lemma 4.1.3. *Let $q = p^m$ be a prime power, and $J/\mathbb{F}_q$ be an abelian variety. Let $r \neq p$ be a prime dividing $\#J(\mathbb{F}_q)$, and let k, s be integers not divisible by r. Then*

1. *J has embedding degree k with respect to r if and only if $r \mid \Phi_k(q)$.*

2. *The minimal embedding field of J with respect to r is $\mathbb{F}_{p^s}$ if and only if $r \mid \Phi_s(p)$.*

Proof. The first statement appears e.g. as [43, Proposition 2.4]; we observe that the same proof applies to the second statement. □

Lemma 4.1.3 allows us to rephrase the question of knowing when the minimal embedding field is not a proper subfield of $\mathbb{F}_{q^k}$ into examining whether or not a given r dividing $\Phi_k(p^m)$ does also divide $\Phi_{km}(p)$. To answer the question in this form some extra properties of cyclotomic polynomials are required.

Fact 4.1.4. Let $\Phi_k(x)$ denote the kth cyclotomic polynomial. Then

1. $x^k - 1 = \prod_{d|k} \Phi_d(x)$.
2. The degree of $\Phi_k(x)$ is $\varphi(k) := \#\{e \in \mathbb{Z} : 1 \leq e \leq k \text{ and } \gcd(e,k) = 1\}$.
3. If ℓ is a prime not dividing k, then $\Phi_k(x^\ell) = \Phi_{k\ell}(x)\Phi_k(x)$.
4. If ℓ is a prime dividing k, then $\Phi_k(x^\ell) = \Phi_{k\ell}(x)$.

These properties either appear, or can be easily derived from the discussion of [81, §VI.3], allowing us to prove the following lemma.

Lemma 4.1.5. *If k and m are coprime, then*

$$\Phi_k(x^m) = \prod_{d|m} \Phi_{kd}(x). \tag{4.1.1}$$

Proof. We first compare the degrees of the polynomials on each side of (4.1.1). Clearly the left hand side has degree $m\varphi(k)$. Now for any coprime numbers x and y we have $\varphi(xy) = \varphi(x)\varphi(y)$. Since $(k,m) = 1$ by assumption it is also true that $(k,d) = 1$ for all $d \mid m$. It follows that the degree of the right hand side of (4.1.1) is $\varphi(k)\sum_{d|m}\varphi(d)$, which by Fact 4.1.4 (1) and (2) is equal to $m\varphi(k)$.

We next compare the roots of the two polynomials. First, we observe that by Fact 4.1.4 (1) the right hand side divides $x^{km} - 1$ and thus has only simple roots. Now

suppose ζ is a root of $\Phi_{kd}(x)$ for some $d \mid m$. Since ζ is a primitive kdth root of unity, ζ^d is a primitive kth root of unity. Write $m = de$. Since $\gcd(k, e) = 1$, it follows that $(\zeta^d)^e = \zeta^m$ is also a primitive kth root of unity, so ζ is also a root of $\Phi_k(x^m)$.

Since the two polynomials in (4.1.1) are both monic and have the same degree, and furthermore all roots of the right hand side are simple and are also roots of the left hand side, we conclude that the two polynomials are equal. □

We will now present the main result of this chapter which, although stated in terms of cylotomic polynomials, will allow us to give an answer to our main question.

Theorem 4.1.6. Let k be a positive integer, p^m a prime power, and r a prime. Write $m = \alpha\beta$, where every prime dividing α also divides k and $\gcd(k, \beta) = 1$. (This factorization is unique.) Denote by e the smallest prime factor of β. Suppose $r \mid \Phi_k(p^m)$ and that one of the following holds:

1. $m = \alpha$ (and $\beta = 1$);
2. β is prime and $r > \Phi_{k\alpha}(p)$;
3. $r > p^{km/e}$; or
4. $4 \mid m$ or $2 \mid k$, and $r > p^{km/2e} + 1$.

Then $r \mid \Phi_{km}(p)$.

Proof. We first note that Fact 4.1.4 (4) implies

$$\Phi_k(p^m) = \Phi_{k\alpha}(p^\beta). \tag{4.1.2}$$

Since $k\alpha$ and β are coprime, Lemma 4.1.5 implies that $\Phi_k(p^m)$ has $\Phi_{km}(p)$ as a factor. Our strategy in each case is to show that the remaining factors of $\Phi_k(p^m)$ are all smaller than r. Since r is prime, it then follows that if r divides $\Phi_k(p^m)$ then r divides $\Phi_{km}(p)$.

We now consider each case separately:

1. Since $m = \alpha$ it follows immediately that $\Phi_k(p^m) = \Phi_{km}(p)$.

2. Since β is a prime not dividing $k\alpha$, equation (4.1.2) and fact 4.1.4 (3) imply that

$$\Phi_k(p^m) = \Phi_{k\alpha\beta}(p)\Phi_{k\alpha}(p) = \Phi_{km}(p)\Phi_{k\alpha}(p).$$

   Since $r > \Phi_{k\alpha}(p)$, it follows that $r \mid \Phi_{km}(p)$.

3. By equation (4.1.2) and Lemma 4.1.5 we have

$$\Phi_k(p^m) = \prod_{d|\beta} \Phi_{kd\alpha}(p) = \prod_{d|\beta} \Phi_{km/d}(p). \tag{4.1.3}$$

   By assumption we have $r > p^{km/d}$ for all $d \mid \beta$ except for $d = 1$, and by Fact 4.1.4 (1) we have $p^{km/d} > \Phi_{km/d}(p)$ for all such d. It follows that $r \mid \Phi_{km}(p)$.

4. Given the factorization of $\Phi_k(p^m)$ as in (4.1.3), the same analysis as in Case 3 shows that $r > \Phi_{km/d}(p)$ for all $d \mid \beta$ with $d \geq 2e$. Since e is the smallest prime dividing β, if $d \mid \beta$ and $1 < d < 2e$ then d is prime, so it suffices to show that $r > \Phi_{km/d}(p)$ for all primes d dividing β. Let d be such a prime. The assumption $4 \mid m$ or $2 \mid k$ then implies that km/d is even. In this case we have $x^{km/d} - 1 = (x^{km/2d} + 1)(x^{km/2d} - 1)$, and $\Phi_{km/d}(x)$ must divide the first factor by Fact 4.1.4 (1). Since $d \geq e$, if $r > p^{km/2e} + 1$ then $r > \Phi_{km/d}(p)$. □

In order to link this result on cyclotomic polynomials to abelian varieties, we use lemma 4.1.3, leading to the following corollary:

Corollary 4.1.7. *Let J be an abelian variety over $\mathbb{F}_q$, where $q = p^m$ with p prime. Let $r \neq p$ be a prime dividing $\#J(\mathbb{F}_q)$, and suppose J has embedding degree k with respect to r. Assume that $r \nmid km$. If q, k, and r satisfy any of the conditions* (1)–(4) *of Theorem 4.1.6, then the minimal embedding field of J with respect to r is $\mathbb{F}_{p^{km}}$.*

We note that if m is prime, usually $r \approx p^{mg}$, with $g = \dim J$, and $m \gg k$, then case (2) of Theorem 4.1.6 applies. Another situation of interest is when p is small, $p = 2$ or $p = 3$ are common choices, then the bound on r given by the theorem is very weak, i.e. J will have minimal embedding field $\mathbb{F}_{q^k}$ with respect to any r used in practice.

It is interesting to note that, although the way theorem 4.1.6 is stated should allow its application to abelian varieties over finite fields that are not pairing-friendly, some cases remain unanswered. For instance, if $k \gg m$ and the dimension g is small then none of the conditions of theorem 4.1.6 can be expected to hold: condition (1) is very unlikely and conditions (2)–(4) would require $r \gg q^g$, which is impossible.

Moreover we remark that, if k is odd and m is even then $\Phi_k(x^m) = \Phi_k(x^{m/2})\Phi_{2k}(x^{m/2})$. Since $\varphi(k) = \varphi(2k)$ for odd k, these two factors have the same degree and the above techniques cannot be used to show that r divides $\Phi_{km}(p)$ and does not divide $\Phi_{km/2}(p)$. Applying theorem 4.1.6 recursively to each factor allows us to determine conditions on q, k, and r guaranteeing that r divides one of the two expressions $\Phi_{km}(p)$ and $\Phi_{km/2}(p)$, but some additional information is required to determine which one.

In the context of pairing-friendly curves, this situation rarely occurs as even embedding degrees and prime values for m are preferred in practice. However, when this situation arise it has to be solved on a case by case basis, as done in propositions 4.2.4 and 4.2.5 below.

4.2 Supersingular elliptic curves over extension fields

Supersingular elliptic curves, are the most well known pairing-friendly abelian varieties defined over non-prime fields, and as such are often used. Usually, in order to optimize implementation, when using a supersingular curve, the curve is chosen to have the maximal embedding degree, that is, a supersingular curve over $\mathbb{F}_{2^m}$ with embedding degree $k = 4$ or over $\mathbb{F}_{3^m}$ with embedding degree $k = 6$. Such curves mostly have

near-prime order, i.e. their order can be written as a product of a large prime and a small factor, and, as defined over field of small characteristic, benefit from some curve arithmetic optimization. This makes them a really good choice for efficient pairing implementations, especially because their minimal embedding field is $\mathbb{F}_{q^k}$ as stated in the two following propositions.

Proposition 4.2.1 ($k = 4$)**.** *Let $q = 2^m$ with m odd, and let E be a supersingular elliptic curve over $\mathbb{F}_q$ that has embedding degree 4 with respect to a prime $r \nmid 2m$. If either*

- $\rho < \frac{3}{2}\left(1 - \frac{1}{\log_2 r}\right)$, *or*
- *m is prime and $r > 5$,*

then E has minimal embedding field $\mathbb{F}_{q^4}$.

Proof. If we write $m = \alpha\beta$ as in Theorem 4.1.6, then the smallest prime dividing β must be at least 3. Thus if $r > q^{2/3} + 1$ then condition (4) of Theorem 4.1.6 is satisfied. If m is prime and $r > 5 = \Phi_4(2)$ then condition (2) of Theorem 4.1.6 is satisfied. In both cases, by Corollary 4.1.7 E has minimal embedding field $\mathbb{F}_{q^4}$. An easy calculation shows that if $\rho < \frac{3}{2}(1 - \frac{1}{\log_2 r})$ then $r > q^{2/3} + 1$. □

Proposition 4.2.2 ($k = 6$)**.** *Let $q = 3^m$ with m odd, and let E be a supersingular elliptic curve over $\mathbb{F}_q$ that has embedding degree 6 with respect to a prime $r \nmid 6m$. If either*

- $\rho < \frac{5}{3}\left(1 - \frac{1}{\log_2 r}\right)$, *or*
- *m is prime and $r > 7$,*

then E has minimal embedding field $\mathbb{F}_{q^6}$.

Proof. The proof is entirely analogous to that of Proposition 4.2.1. □

We remark that, in both of the above cases the exponent c_J defined by Rubin and Silverberg is equal to k. Their result (Theorem 4.1.2) then implies that when $k = 4$, the conclusion of proposition 4.2.1 holds whenever $\rho < \frac{3\log 2}{2\log(1+\sqrt{2})} \approx 1.18$, and that when $k = 6$, the conclusion of proposition 4.2.2 holds whenever $\rho < \frac{3\log 3}{2\log(1+\sqrt{3})} \approx 1.64$. Thus, in both cases our result is stronger, as it requires a weaker upper bound on ρ, for sufficiently large r.

In some special cases one may wish to use supersingular elliptic curves with very small embedding degrees for implemention. We thus continue our analysis by investigating the cases $1 \leq k \leq 3$. The case $k = 2$ is the most straightforward.

Proposition 4.2.3 ($k = 2$)**.** *Let $q = p^m$, and let E be a supersingular elliptic curve over $\mathbb{F}_q$ that has embedding degree 2 with respect to a prime $r \nmid 2m$. If either*

- $\rho < 3\left(1 - \dfrac{1}{\log_2 r}\right)$, *or*
- *m is prime and $r > p + 1$,*

then E has minimal embedding field $\mathbb{F}_{q^2}$.

Proof. The proof is entirely analogous to that of Proposition 4.2.1. □

Rubin and Silverberg's result (theorem 4.1.2) says that the conclusion of Proposition 4.2.3 holds whenever $\rho < 2 - \epsilon$ when m is even and whenever $\rho < 3 - \epsilon$ when m is odd, with $\epsilon \to 0$ as $p \to \infty$. Thus our result is stronger when m is even.

The cases $k = 1$ and $k = 3$ are more subtle, as it is not really possible to avoid the minimal embedding field to be $\mathbb{F}_{q^{k/2}}$ even when r is very large. However, if the sign of the trace is known, then theorem 4.1.6 can be applied to determine when the minimal embedding field is $\mathbb{F}_{q^k}$ or $\mathbb{F}_{q^{k/2}}$.

Proposition 4.2.4 ($k = 1$)**.** *Let $q = p^m$ with m even, and let E be a supersingular elliptic curve over $\mathbb{F}_q$ that has embedding degree 1 with respect to a prime $r \nmid m$. If E*

has trace $-2p^{m/2}$ *and* $\rho < 6(1 - \frac{1}{\log_2 r})$*, then* E *has minimal embedding field* $\mathbb{F}_q$*. If* E *has trace* $2p^{m/2}$ *and* $\rho < 4$*, then* E *has minimal embedding field* $\mathbb{F}_{q^{1/2}}$*.*

Proof. Let $m' = m/2$. Suppose E has trace $-2p^{m'}$. Then $\#E(\mathbb{F}_q) = (p^{m'} + 1)^2$, so r divides $\Phi_2(p^{m'})$. We now apply Theorem 4.1.6 with $k = 2$ and $m = m'$. If we write $m' = \alpha\beta$ as in the theorem, then the smallest prime dividing the β of theorem 4.1.6 must be at least 3. Thus if $r > p^{m'/3} + 1 = q^{1/6} + 1$ then condition (4) of the theorem is satisfied, so by corollary 4.1.7 E has minimal embedding field $\mathbb{F}_{p^{2m'}} = \mathbb{F}_q$. An easy calculation shows that if $\rho < 6(1 - \frac{1}{\log_2 r})$ then $r > q^{1/6} + 1$.

Now suppose E has trace $2p^{m'}$. Then $\#E(\mathbb{F}_q) = (p^{m'} - 1)^2$, so r divides $\Phi_1(p^{m'})$. We now apply theorem 4.1.6 with $k = 1$ and $m = m'$. If $r > p^{m'/2} = q^{1/4}$ (or equivalently, if $\rho < 4$) then condition (3) of the theorem is satisfied, so by corollary 4.1.7 E has minimal embedding field $\mathbb{F}_{p^{m'}} = \mathbb{F}_{q^{1/2}}$. □

When $k = 1$, Rubin and Silverberg's exponent c_J is equal to 1 if E has negative trace and 1/2 if E has positive trace. In both cases the integer f of theorem 4.1.2 is equal to 2. Thus theorem 4.1.2 says that the conclusion of proposition 4.2.4 holds whenever $\rho < 4 - \epsilon$, with $\epsilon \to 0$ as $p \to \infty$. Our result is then stronger for the first case as well as for small p.

In fact, proposition 4.2.4 demonstrates the fact that the minimal embedding field of an elliptic curve E can be smaller than its field of definition. One can construct such a curve as follows: Let $p > 3$ be prime, and let $E/\mathbb{F}_p$ be a supersingular elliptic curve over $\mathbb{F}_p$. If we define $E'/\mathbb{F}_{p^2}$ as a quadratic twist of E over $\mathbb{F}_{p^2}$, then $\#E'(\mathbb{F}_{p^2}) = (p-1)^2$, and the minimal embedding field of E' with respect to any $r \mid p - 1$ is $\mathbb{F}_p$.

Finally, we consider the case of embedding degree $k = 3$. As with $k = 1$, the minimal embedding field can be determined from the sign of the trace.

Proposition 4.2.5 ($k = 3$)**.** *Let* $q = p^m$ *with* m *even, and let* E *be a supersingular elliptic curve over* $\mathbb{F}_q$ *that has embedding degree* 3 *with respect to a prime* $r \nmid 3m$*. If* E

has trace $p^{m/2}$ *and* $\rho < \frac{10}{3}(1 - \frac{1}{\log_2 r})$*, then* E *has minimal embedding field* $\mathbb{F}_{q^3}$*. If* E *has trace* $-p^{m/2}$ *and* $\rho < 4/3$*, then* E *has minimal embedding field* $\mathbb{F}_{q^{3/2}}$*.*

Proof. The proof is entirely analogous to that of Proposition 4.2.4. □

When $k = 3$, Rubin and Silverberg's exponent c_J is equal to 3 if E has positive trace and 3/2 if E has negative trace. Thus theorem 4.1.2 says that the conclusion of proposition 4.2.5 holds whenever $\rho < 2 - \epsilon$, with $\epsilon \to 0$ as $p \to \infty$. Our result is stronger for the first case.

4.3 Higher-dimensional supersingular abelian varieties

In this section we briefly sketch the application of the main result to supersingular abelian varieties of dimension $g \geq 2$ defined over non-prime fields.

We first consider simple supersingular abelian varieties of dimension $g = 2$. Such varieties, known as abelian surfaces, can be described as Jacobians of genus 2 curves. Cardona and Nart [22] give a detailed description of the possible group orders and embedding degrees for simple supersingular abelian surfaces, analogous to the Menezes-Okamoto-Vanstone classification for elliptic curves.

Table 4.1 lists the isogeny classes of simple supersingular abelian surfaces over $\mathbb{F}_q$ and their respective embedding degree k, as determined by Cardona and Nart. The isogeny classes are described by a pair of integers (s, t), which correspond to the coefficients of the characteristic polynomial of Frobenius $x^4 + sx^3 + tx^2 + sqx + q^2$. An asterisk next to the embedding degree indicates that the minimal embedding field is $\mathbb{F}_{q^{k/2}}$, not $\mathbb{F}_{q^k}$.

When the extension degree m is prime, as is most often the case in practice, corollary 4.1.7 tells us that if $r > \Phi_k(p)$ then the minimal embedding field of a supersingular abelian surface with respect to r is $\mathbb{F}_{p^k}$. For the cases of small characteristic, we have the following result.

Proposition 4.3.1. *Let J be a simple supersingular abelian surface over $\mathbb{F}_q$, where $q = p^m$, $p \in \{2, 3, 5\}$, and m is prime. Suppose J has embedding degree k with respect to a prime $r > m$. If $r > 781$ then the minimal embedding field of J with respect to r is $\mathbb{F}_{q^k}$.*

For more general situations, table 4.1 gives two parameters for each isogeny class that are related to the minimal embedding field. A value of a in the column "Cor. 4.1.7 max ρ" indicates that whenever $r \nmid km$ is prime and $\rho < a$, corollary 4.1.7 implies that an abelian variety in the isogeny class has minimal embedding field equal to either $\mathbb{F}_{q^k}$ with respect to r, or $\mathbb{F}_{q^{k/2}}$ in the asterisked cases. When the value is $a - \epsilon$ one can take $\epsilon = a/\log_2 r$.

A value of b in the column "RS max ρ" indicates that whenever r is prime and $\rho < b$, Rubin and Silverberg's result (theorem 4.1.2) implies that an abelian variety in the isogeny class has minimal embedding field $\mathbb{F}_{q^k}$ with respect to r (or $\mathbb{F}_{q^{k/2}}$ in the asterisked cases). When p is not fixed, the values b are limits as $p \to \infty$.

(s, t)	conditions on p and m	k	Cor. 4.1.7 max ρ	RS max ρ
$(0, -2q)$	m odd	1	6	6
$(0, 2q)$	m even, $p \equiv 1 \pmod 4$	2	$6 - \epsilon$	4
$(2\sqrt{q}, 3q)$	m even, $p \equiv 1 \pmod 3$	3*	$8/3$	4
$(-2\sqrt{q}, 3q)$	m even, $p \equiv 1 \pmod 3$	3	$20/3 - \epsilon$	4
$(0, 0)$	m odd, $p \neq 2$	4	$3 - \epsilon$	3
$(0, 0)$	m even, $p \not\equiv 1 \pmod 8$	4	$3 - \epsilon$	2
$(0, q)$	m odd	3	$10/3$	3
$(0, -q)$	m odd, $p \neq 3$	6	$10/3 - \epsilon$	3
$(0, -q)$	m even, $p \not\equiv 1 \pmod{12}$	6	$10/3 - \epsilon$	2
$(\sqrt{q}, q)$	m even, $p \not\equiv 1 \pmod 5$	5*	$8/5$	2
$(-\sqrt{q}, q)$	m even, $p \not\equiv 1 \pmod 5$	5	$12/5 - \epsilon$	2
$(\pm\sqrt{5q}, 3q)$	m odd, $p = 5$	5	$6/5$	2.06
$(\pm\sqrt{2q}, q)$	m odd, $p = 2$	12	$5/3 - \epsilon$	1.18

Table 4.1: Isogeny classes of simple supersingular abelian surfaces over $\mathbb{F}_q$.

An interesting situation to analyse is the case of supersingular abelian varieties of

dimension $g = 4$. Rubin and Silverberg [112, §5.1] showed that if $q = 3^m$ and E is a supersingular elliptic curve over $\mathbb{F}_q$ with embedding degree 6, then there is a simple 4-dimensional abelian variety $J/\mathbb{F}_q$ with embedding degree $k = 30$. This J can be constructed as a subvariety of the restriction of scalars $\mathrm{Res}_{\mathbb{F}_{q^5}/\mathbb{F}_q} E$.

Proposition 4.3.2. *Let $q = 3^m$ with m odd, and let A be a simple supersingular 4-dimensional abelian variety over $\mathbb{F}_q$ that has embedding degree 30 with respect to a prime $r \nmid 30m$. If either*

- $\rho < \dfrac{28}{15}\left(1 - \dfrac{1}{\log_2 r}\right)$, *or*
- *m is prime and $r > 8400$,*

then A has minimal embedding field $\mathbb{F}_{q^{30}}$.

Proof. The proof is entirely analogous to that of Proposition 4.2.1. □

We note that if J is an abelian variety as in proposition 4.3.2, Rubin and Silverberg's result (pheorem 4.1.2) shows that the result holds whenever $r > (1 + \sqrt{3})^{8m/3}$, or $\rho \lessapprox 1.64$. Thus our result ($\rho \lessapprox 1.87$) is stronger.

4.4 Discussion

For an abelian variety J defined over a finite field $\mathbb{F}_q$ such that J has embedding degree k with respect to a subgroup of prime order r, the question of knowing whether or not the minimal embedding field of J with respect to r is $\mathbb{F}_{q^k}$ can be answered in terms of q, r and k under certain conditions expressed in theorem 4.1.6 and corollary 4.1.7.

When theorem 4.1.6 is applied to supersingular elliptic curves (section 4.2) and to supersingular genus 2 curves (section 4.3), by computing a maximum ρ-value for which the minimal embedding field must be $\mathbb{F}_{q^k}$, it, most of the time, results in larger allowable ρ-values than the corresponding result of Rubin and Silverberg (theorem 4.1.2).

Another interesting result is that theorem 4.1.6 holds for general abelian varieties, not only supersingular ones. Several results demonstrate the existence of non-supersingular abelian varieties over extension fields with small embedding degree [48, 69], but at present only a single explicit construction of such varieties has been exhibited. This construction, due to Hitt O'Connor, McGuire, Naehrig and Streng [70, Algorithm 3], produces abelian surfaces over $\mathbb{F}_{p^2}$ with p-rank 1, i.e. neither ordinary nor supersingular, and $\rho \approx 16$. These ρ-values are far too large both for practical use and for Corollary 4.1.7 to provide any useful result.

The construction of non-supersingular abelian varieties over non-prime fields with small embedding degree and $\rho < 16$ is still an open-problem. Finding such varieties would not only expand the library of pairing-friendly abelian varieties but could potentially lead to different improvements in practice. In this case the results presented in this chapter could be used in order to describe the minimal embedding field of these varieties.

As efficiency is one of the major concerns when it comes to implementing and using pairings we will now focus in the next chapter on how to improve their computational speed.

5

Pairings and efficiency

> In all affairs it's a healthy thing now and then to hang a question mark on the things you have long taken for granted.
>
> B. Russell

Using the important techniques listed in chapter 3 (section 3.3), improves Miller's algorithm. We will now consider closely how to handle the fifth one, to efficiently compute the final exponentiation.

5.1 The final exponentiation

After the Miller loop the Tate pairing carries out an extra step to ensure a unique result of the pairing, as f must be raised to be power $(p^k - 1)/r$. Since p, k and r are fixed system parameters it is possible to optimise the so-called final exponentiation.

We start by restricting our attention to the case of even embedding degrees, which are more useful and practical, as they support the important *denominator elimination* optimization [8]. Thus the final exponent can be broken down into three components. Let $d = k/2$. Then

$$(p^k - 1)/r = (p^d - 1) \cdot [(p^d + 1)/\Phi_k(p)] \cdot [\Phi_k(p)/r].$$

The field characteristic being p, the first two parts of the exponentiation only consists of applying the Frobenius operator [14] in order to raise to the power of p. This results in an almost free computation, as such, the first two parts are called *easy*. More than being cheap, although it requires an extension field division, the first part of the exponentiation simplifies the rest of the final exponentiation. After raising to the power of $(p^d - 1)$ the field element becomes *unitary* [122], i.e. an element α with norm $N_{\mathbb{F}_{p^k}/\mathbb{F}_{p^d}}(\alpha) = 1$. This has important implications, as squaring of unitary elements is significantly cheaper than squaring of non-unitary elements, and any future inversions can be implemented by simple conjugation [129], [122], [61], [99].

Once the easy part is computed the *hard part* of the final exponentiation still remains, that is, raising to the power of $\Phi_k(p)/r$. This is usually done by expressing this exponent to the base p as $\lambda_{n-1} \cdot p^{n-1} + ... + \lambda_1 \cdot p + \lambda_0$, where $n = \varphi(k)$. If the value to be exponentiated is m, then we need to calculate

$$m^{\lambda_{n-1} \cdot p^{n-1}} .... m^{\lambda_1 \cdot p} \cdot m^{\lambda_0},$$

which can be rewritten

$$(m^{p^{n-1}})^{\lambda_n - 1} ..... (m^p)^{\lambda_1} \cdot m^{\lambda_0}.$$

The m^{p^i} can be calculated using the Frobenius, and the hard part of the final exponentiation can be computed using a fast multi-exponentiation algorithm [66], [56], [92].

However, doing so does not take advantage of the form of the polynomial describing p and r. We will now present, for some families of pairing-friendly elliptic curve having ρ-value close to 1, a new method benefiting from the construction of the curves in order to efficiently compute the hard part of the exponentiation.

5.1.1 MNT curves

As recalled in chapter 3, MNT pairing-friendly elliptic curves with embedding degree $k = 6$ can be parameterised using the following polynomials:

$$\begin{aligned} t(x) &= x + 1 \\ r(x) &= x^2 - x + 1 \\ p(x) &= x^2 + 1. \end{aligned}$$

In this case the hard part of the final exponentiation is $(p^2 - p + 1)/r$. Substituting p and r by their respective corresponding polynomials from above leads to $(x^4 + x^2 + 1)/(x^2 - x + 1) = x^2 + x + 1$. By expressing it to the base p, it becomes $(p + x)$ and the hard part of the final exponentiation is $m^p.m^x$. This is done by only using an application of the Frobenius and an exponentiation to the power of x. The advantage of deriving the hard part of the exponentiation in terms of the family parameter x is clearly illustrated in this simple case, as x is only half the size of p.

5.1.2 BN curves

Pairing-friendly elliptic curves from the BN family have embedding degree 12, and can be parameterised as follows:

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

In this case the hard part of the final exponentiation is to the power of $(p^4 - p^2 + 1)/r$. After substituting the polynomials for p and r this can be expressed to the base p as

$$\lambda_3.p^3 + \lambda_2.p^2 + \lambda_1.p + \lambda_0,$$

where

$$\begin{aligned}
\lambda_3(x) &= 1; \\
\lambda_2(x) &= 6x^2 + 1; \\
\lambda_1(x) &= -36x^3 - 18x^2 - 12x + 1; \\
\lambda_0(x) &= -36x^3 - 30x^2 - 18x - 2.
\end{aligned}$$

Although this expression is more complex than in the case of MNT curves it is still possible to handle it efficiently taking a new approach. BN curves being very plentiful, it is possible to choose x with low Hamming weight. The resulting polynomial $r(x)$ then, has low Hamming weight, allowing a faster computation of the Miller loop. The next stage is the computation of m^x, $m^{x^2} = (m^x)^x$ and $m^{x^3} = (m^{x^2})^x$. These are simple exponentiations, and the low Hamming weight of x ensures that each one of them requires a minimum number of multiplications when using a simple square-and-multiply algorithm. Then computing m^p, m^{p^2}, m^{p^3}, $(m^x)^p$, $(m^{x^2})^p$, $(m^{x^3})^p$ and $(m^{x^2})^{p^2}$ can be done efficiently by using the Frobenius. If we group the elements of the exponentiation together, the expression becomes:

$$[m^p \cdot m^{p^2} \cdot m^{p^3}] \cdot [1/m]^2 \cdot [(m^{x^2})^{p^2}]^6 \cdot [(m^x)^p]^{12} \cdot [m^x/((m^{x^2})^p)]^{18} \cdot [1/m^{x^2}]^{30} \cdot [m^{x^3} \cdot (m^{x^3})^p]^{36}.$$

Recalling that division costs the same as multiplication, as inversion is just a conjugation for unitary elements, the individual components between the square brackets can be calculated using only four multiplications. This leaves us with a calculation of

the form:

$$y_0 \cdot {y_1}^2 \cdot {y_2}^6 \cdot {y_3}^{12} \cdot {y_4}^{18} \cdot {y_5}^{30} \cdot {y_6}^{36}. \tag{5.1.1}$$

In fact, the exponents in this expression are simply the coefficients that arise in the λ_i equations. Thus, the initial question, of knowing how to efficiently compute the hard part of the exponentiation, boils down to how best to evaluate the above product.

The goal being the minimization of the number of multiplications, Olivos' algorithm [104] [7, Section 9.2] is perfectly suited to this case. Given a number n, we define an *addition chain* as a set of integers such that each element can be written as the sum of two previous elements, the first element of the chain being 1 and the last one being n. When a set S is given, instead of a number, the resulting chain, including all the elements of S, is called an *addition sequence*. The idea behind Olivos's algorithm is to consider all the exponents as a set and return the corresponding shortest addition sequence. This results in an optimal decomposition of the exponentiations. When this strategy is applied to equation 5.1.1, it leads to the following addition sequence:

$$\{1, 2, 6, 12, 18, 30, 36\} \rightarrow \{1, 2, \underline{3}, 6, 12, 18, 30, 36\}.$$

To obtain a proper addition sequence we see that 3 must be added to the initial set. This is the only element not belonging to the set of exponents, which means less work to do the evaluation.

In our case Olivos' algorithm consists in considering the vectors $Y_i = (0, \cdots, 1, \cdots, 0)$, the i-th component of the vector being 1, and apply operations on the vectors such that it yields the vector $(36, 30, 18, 12, 6, 2, 1)$. This leads to the "vectorial addition chain" given in table 5.1. In turn, it allows the evaluation of expression 5.1.1 using just two temporary variables, T_0 and T_1, as described in algorithm 5.1. This part of the calculation requires only 9 multiplications and 4 squaring.

If we take the low hamming weight value $x = -4080000000000001_{16}$ suggested

$(y_6$	y_5	y_4	y_3	y_2	y_1	$y_0)$
(1	0	0	0	0	0	0)
(0	1	0	0	0	0	0)
(0	0	1	0	0	0	0)
(0	0	0	1	0	0	0)
(0	0	0	0	1	0	0)
(0	0	0	0	0	1	0)
(0	0	0	0	0	0	1)
(2	0	0	0	0	0	0)
(2	0	1	0	0	0	0)
(2	1	1	0	0	0	0)
(0	1	0	1	0	0	0)
(2	2	1	1	0	0	0)
(2	1	1	0	1	0	0)
(4	4	2	2	0	0	0)
(6	5	3	2	1	0	0)
(12	10	6	4	2	0	0)
(12	10	6	4	2	1	0)
(12	10	6	4	2	0	1)
(24	20	12	8	4	2	0)
(36	30	18	12	6	2	1)

Table 5.1: Olivos' algorithm in the case of BN curves.

by Nogami, Akane, Sakemi, Kato and Morikawa in [102], it lowers the number of multiplications/squarings over $\mathbb{F}_p$ from 7426 to 7156, which, in practice, represents a 4% speed increase. Hence, this new approach to the hard part of the final exponentiation leads to significant efficiency improvement, in the case of BN curves, but not only those as will see next.

5.1.3 Freeman Curves

Freeman suggested the construction of pairing-friendly elliptic curves of embedding degree 10, using the following parameters to describe the family:

Algorithm 5.1 Evaluation of expression 5.1.1 using only two temporary variables.

INPUT: $y_0, y_1, y_2, y_3, y_4, y_5, y_6$.
OUTPUT: $y_0 \cdot {y_1}^2 \cdot {y_2}^6 \cdot {y_3}^{12} \cdot {y_4}^{18} \cdot {y_5}^{30} \cdot {y_6}^{36}$.

1: $T_0 \leftarrow (y_6)^2$
2: $T_0 \leftarrow T_0 \cdot y_4$
3: $T_0 \leftarrow T_0 \cdot y_5$
4: $T_1 \leftarrow y_3 \cdot y_5$
5: $T_1 \leftarrow T_1 \cdot T_0$
6: $T_0 \leftarrow T_0 \cdot y_2$
7: $T_1 \leftarrow (T_1)^2$
8: $T_1 \leftarrow T_1 \cdot T_0$
9: $T_1 \leftarrow (T_1)^2$
10: $T_0 \leftarrow T_1 \cdot y_1$
11: $T_1 \leftarrow T_1 \cdot y_0$
12: $T_0 \leftarrow (T_0)^2$
13: $T_0 \leftarrow T_0 \cdot T_1$
14: **return** T_0

$$\begin{aligned} t(x) &= 10x^2 + 5x + 3 \\ r(x) &= 25x^4 + 25x^3 + 15x^2 + 5x + 1 \\ p(x) &= 25x^4 + 25x^3 + 25x^2 + 10x + 3. \end{aligned}$$

These curves are much rarer than the BN curves, and unfortunately it is not feasible to choose x to have a particularly small Hamming weight. Nevertheless proceeding as above is still possible:

$$\begin{aligned} \lambda_3(x) &= 1; \\ \lambda_2(x) &= 10x^2 + 5x + 5; \\ \lambda_1(x) &= -5x^2 - 5x - 3; \\ \lambda_0(x) &= -25x^3 - 15x^2 - 15x - 2. \end{aligned}$$

In this case the coefficients form a perfect addition chain, i.e. no elements need to be added:

$$\{1, 2, 3, 5, 10, 15, 25\}.$$

The optimal vectorial addition chain in this case requires 10 multiplications and 2 squarings.

5.1.4 KSS Curves

The Kachisa, Schaeffer and Scott method leads to few families with different embedding degrees. We will consider the families with embedding degree $k = 8$ and $k = 18$.

KSS curves ($k = 8$): The parameters for this family are given by:

$$\begin{aligned} t(x) &= \frac{1}{15}(2x^3 - 11x + 15) \\ r(x) &= \frac{1}{450}(x^4 - 8x^2 + 25) \\ p(x) &= \frac{1}{180}(x^6 + 2x^5 - 3x^4 + 8x^3 - 15x^2 - 82x + 125). \end{aligned}$$

We note that, as BN curves, these curves are plentiful, and then x can be chosen to have a low Hamming weight. The decomposition of the hard part to base p yields:

$$\begin{aligned} \lambda_3(x) &= \frac{1}{6}(15x^2 + 30x + 75) \\ \lambda_2(x) &= \frac{1}{6}(2x^5 + 4x^4 - x^3 + 26x^2 - 55x - 144) \\ \lambda_1(x) &= \frac{1}{6}(-5x^4 - 10x^3 - 5x^2 - 80x + 100) \\ \lambda_0(x) &= \frac{1}{6}(x^5 + 2x^4 + 7x^3 + 28x^2 + 10x + 108). \end{aligned}$$

The major difference compared to previous cases is in the common denominator 6 appearing for each λ_i. Since in practice r will be large and coprime to 6, this issue can easily be overcome by evaluating the sixth power of the pairing instead of the pairing

itself. Thus the result of the exponentiation will still belong to a group of order r and it suffices to simply ignore the denominator. It results in the following optimal addition sequence which contains all the exponents in the above equations:

$$\{1, 2, 4, 5, 7, 10, 15, \underline{25}, 26, 28, 30, \underline{36}, \underline{50}, 55, 75, 80, 100, 108, 144\}.$$

The underlined numbers are the extra numbers added in order to complete the sequence. Proceeding as in the BN case, the vectorial addition chain derived from this addition sequence requires only 27 multiplications and 6 squarings to complete the calculation of the hard part of the final exponentiation.

KSS curves ($k = 18$): This family is defined by the following polynomials:

$$\begin{aligned}
t(x) &= \frac{1}{7}(x^4 + 16x + 7) \\
r(x) &= \frac{1}{343}(x^6 + 37x^3 + 343) \\
p(x) &= \frac{1}{21}(x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401).
\end{aligned}$$

Although, as recalled in chapter 3 section 3.2, $t(x)$, $r(x)$ and $p(x)$ evaluate as integers if $x \equiv 14 \mod 42$, x can still be chosen with a low Hamming weight. Then by proceeding as usual we find:

$$\begin{aligned}
\lambda_5(x) &= \frac{1}{3}(49x^2 + 245x + 343) \\
\lambda_4(x) &= \frac{1}{3}(7x^6 + 35x^5 + 49x^4 + 112x^3 + 581x^2 + 784x) \\
\lambda_3(x) &= \frac{1}{3}(-5x^7 - 25x^6 - 35x^5 - 87x^4 - 450x^3 - 609x^2 + 54) \\
\lambda_2(x) &= \frac{1}{3}(-49x^5 - 245x^4 - 343x^3 - 931x^2 - 4802x - 6517) \\
\lambda_1(x) &= \frac{1}{3}(14x^6 + 70x^5 + 98x^4 + 273x^3 + 1407x^2 + 1911x) \\
\lambda_0(x) &= \frac{1}{3}(-3x^7 - 15x^6 - 21x^5 - 62x^4 - 319x^3 - 434x^2 + 3).
\end{aligned}$$

Using the same argument as in the KSS $k = 8$ curves case, we evaluate the cube of the pairing to remove the awkward denominator of 3. In this case the coefficients again "nearly" form a natural addition sequence. A relatively short addition sequence containing all of the exponents in the above λ_i, is:

$$\{\underline{1}, \underline{2}, 3, \underline{4}, 5, 7, \underline{8}, 14, 15, \underline{16}, 21, 25, \underline{28}, 35, \underline{42}, 49, 54, 62, 70, 87, 98, 112, \underline{147}, 245, 273, \underline{294}$$

$$319, 343, \underline{392}, 434, 450, 581, 609, 784, 931, \underline{1162}, 1407, \underline{1862}, 1911, \underline{3724}, \underline{4655}, 4802, 6517\}.$$

It is interesting to note that, in this case, it is feasible to find a shorter addition chain. However, if we take into consideration the fact that squaring is notably cheaper than multiplication over an extension field, it may happen that a longer chain gives rise to a more efficient computation. Here, it requires 56 multiplications and 14 squarings, instead of 61 multiplications and only 7 squarings in order to complete the calculation of the hard part of the final exponentiation. Hence, it can happen that slightly longer sequences are preferable to shorter ones if it features more doubling and less additions which, in turn, results in more squarings and less multiplications.

5.2 Discussion

One of the first remarks concerns the hardness of finding the shortest addition sequence. In fact, it is an $\mathcal{NP}$-complete problem [34], but since the values we obtained in each set are relatively small, and the sets themselves already contained some addition "sub-chains", it is, in this context, not too difficult to generate, either with a computer or manually, addition sequences containing the specific entries with length close to the lower bound given for the length of addition chains [18]. Should a particular curve result in larger or more numerous coefficients to be constructed into a sequence, Bos and Coster suggest an algorithm for that scenario in [18].

An other important remark is related to the length of the chain. In fact, since squarings are significantly faster than multiplications over extension fields, it may, as we have seen, be sometimes preferable to select a slightly longer addition sequence which trades additions for doublings. From an efficiency point of view, the unitary property implies that divisions are not more expensive that multiplications, rendering addition-subtraction chains a good option for more complicated expressions. This would result in "unordered" sequences.

On the sequences themselves, it is interesting to note their compactness, implying that really few values need to be added to the coefficient in the λ_i. Those coefficients also feature the special property of having relatively small factors, tending to be "smooth" numbers. This seems to facilitate the construction of addition sequences. In some cases, like the Freeman curves, the coefficient of the λ_i already form an addition sequence, and if we extend the method to BW curves it often leads to addition sequences as easy as:

$$\{1, 2, 3\}.$$

Other intriguing patterns emerge as in the case of the KSS $k = 18$ curves where the three most significant coefficients of the λ_i are all in the same ratio 1:5:7. Coefficients also appear to follow the same kind of distribution as numbers in a typical addition chain.

One of the main benefits of this new method is that it allows the writing of computer programs which automatically generate very efficient pairing code, given only the polynomial equations defining a pairing-friendly family of elliptic curves [33]. This is very much appreciated for practical use as implementing pairings is often hard and requires a good knowledge of a wide range of primitives.

6

Pairings and the discrete logarithm problem

Something convincing is not necessarily true, it is only convincing.

F. Nietzsche

As computing pairings can be done efficiently Menezes, Okamoto and Vanstone [91], had the idea of using them to map a hard problem over an elliptic curve into an easier problem over a finite field. In fact, although their initial aim targeted supersingular elliptic curves, it extends by definition to all ordinary pairing-friendly elliptic curves.

The hard problem of concern, namely the Elliptic Curve Discrete Logarithm Problem (ECDLP), is defined for an elliptic curve E over $\mathbb{F}_q$, P a generator of a subgroup $\mathbb{G}$ of $E(\mathbb{F}_q)$, and $Q \in \mathbb{G}$, as finding an x such that $Q = [x]P$. Using a pairing it is then transformed into the Discrete Logarithm Problem (DLP), which is the equivalent of the ECDLP over a finite field: given α a generator of a subgroup $\mathbb{G}$ of $\mathbb{F}_{q^k}$, and $\beta \in \mathbb{G}$, find x such $\alpha^x = \beta$.

We can immediately and easily note that for the ECDLP to be unsolvable, one must ensure that the DLP is also intractable, if dealing with pairing-friendly elliptic

curves. Indeed, let $Q = [x]P$, with P a point of order r on $E(\mathbb{F}_q) \cong \mathbb{Z}/r\mathbb{Z} \oplus \mathbb{Z}/n\mathbb{Z}$, where $n|r$ and $n|(q-1)$ [91, section II]. If we take a point G on the curve such that (P, G) generates $E(\mathbb{F}_q)$, and a point $S = [s_1]P + [s_2]G$, for some integers s_1, s_2, then

$$\begin{aligned} e_r(P,T)^n &= e_r(P,P)^{s_1 n} e_r(P,[s_2 n]G) \\ &= e_r(P,\mathcal{O}) \\ &= 1 \end{aligned}$$

Hence the order of $e_r(P,T)$ divides n, and as $n|(q-1)$, $e_r(P,T) \in \mathbb{F}_q$. We also have

$$e_r(Q,S) = e_r([x]P,S) = e_r(P,S)^x.$$

Therefore if the pairing can be computed efficiently it is possible to solve the ECDLP, by solving the DLP. Thus in order to clearly state which parameters should be used a more advanced study of the best known algorithms to solve both the ECLP and the DLP is required.

6.1 Theoretical view

From a theoretical point of view how an algorithm performs is based on its complexity. Therefore, in order to know how easy it is to solve the ECDLP and the DLP we will describe and analyse the most efficient algorithms known to date, for solving those two problems.

6.1.1 Pollard's Rho algorithm

Pollard's Rho algorithm [110] is very interesting as it applies to any group, not depending on any specific structure. As such, it applies to elliptic curves, and is in fact the best known general algorithm to solve the ECDLP.

The core idea of Pollard's Rho algorithm relies on the *birthday paradox*, which states

that in a random set of people, the probability to have two persons born on the same date is over 50% as soon as the set contains more than 23 people and reaches 99% with only 57. Adapted to the case of a cyclic group $\mathbb{G}$ of order n, it means that a collision between two elements will occur in time $\sqrt{n}$.

In the case of $\mathbb{G}$ being a multiplicative group, the main goal is to obtain a collision of two elements $\alpha^{a_1}\beta^{b_1} \equiv \alpha^{a_2}\beta^{b_2} \mod n$, which yields

$$\alpha^{\frac{a_2-a_1}{b_1-b_2}} = \beta. \tag{6.1.1}$$

This is achieved by first remarking that for α a generator, all the elements of $\mathbb{G}$ can be written $\alpha^a\beta^b$ for $\beta \in \mathbb{G}$. Then, if a collision occurs between two elements x and y it is sufficient to know their decomposition into an α, β product, and apply the above idea to find a similar equation to 6.1.1.

More formally, it is done by defining S_1, S_2 and S_3, three subsets of $\mathbb{G}$ of approximately the same size, and three functions f, g and h on elements of $\mathbb{G}$ as follows:

$$h(b,x) = \begin{cases} b & x \in S_1 \\ 2b \mod n & x \in S_2 \\ b+1 \mod n & x \in S_3 \end{cases}$$

$$f(x) = \begin{cases} \beta & x \in S_1 \\ \alpha x & x \in S_2 \\ x^2 & x \in S_3 \end{cases} \qquad g(a,x) = \begin{cases} a+1 \mod n & x \in S_1 \\ 2a \mod n & x \in S_2 \\ a & x \in S_3 \end{cases}$$

This being set, it suffices to follow algorithm 6.1 in order to solve the DLP.

The running time is obviously the time required to get a collision, which is as stated above, $O(\sqrt{n})$, for n the size of the group $\mathbb{G}$.

Algorithm 6.1 Pollard's Rho algorithm.

INPUT: α a generator of $\mathbb{G}$ and $\beta \in \mathbb{G}$, $f(x)$, $g(a,x)$ and $h(b,x)$.
OUTPUT: $\log_\alpha \beta$, or failure.

```
a_0 ← 0
b_0 ← 0
x_0 ← 1
i ← 1
repeat
    x_i ← f(x_{i-1})
    a_i ← g(a_{i-1}, x_{i-1})
    b_i ← h(b_{i-1}, x_{i-1})
    x_{2i} ← f(f(x_{2i-2}))
    a_{2i} ← g(g(a_{2i-2}, x_{2i-2}), f(x_{2i-2}))
    b_{2i} ← h(h(b_{2i-2}, x_{2i-2}), f(x_{2i-2}))
    i ← i + 1
until x_i = x_{2i}
r ← b_i − b_{2i}
if r ≠ 0 then
    return r^{-1}(a_{2i} − a_i) mod n
else
    return failed
end if
```

6.1.2 Pohlig Hellman algorithm

A less generic, but more efficient algorithm, relying on the use of the Chinese Remainder Theorem (CRT), was discovered by Pohlig and Hellman [109]. In fact, they realised that, when the order n of the group $\mathbb{G}$ can be factored into small primes it is easy to derive a system of modular equations, which can be solved using the CRT.

More precisely, n is first decomposed into a product of primes $\prod_{i=1}^{t} p_i^{e_i}$, then the core idea is to see that since $x = \log_\alpha \beta$ is unique modulo n, knowing x_i such that $x_i \equiv x \mod p_i^{e_i}$, allows to determine x by only solving a modular system of equations.

Another important point, is that, if x_i is written to base p, $x_i = l_{i,0} + \cdots + l_{i,e_i-1} p^{e_i-1}$, and for each x_i, x is viewed as $x = x_i + sp^{e_i}$ for some integer s, we can consider

$\frac{nx - nl_{i,0}}{p_i}$:

$$\begin{aligned}
\frac{n(x_i + sp_i^{e_i})}{p_i} - \frac{n.l_{i,0}}{p_i} &\equiv \frac{n}{p_i}(x_i + sp_i^{e_i} - l_{i,0}) \mod n \\
&\equiv \frac{n}{p_i}(\sum_{j=0}^{e_i-1} l_{i,j}p_i^j + sp_i^{e_i} - l_{i,0}) \mod n \\
&\equiv \frac{n}{p_i}(\sum_{j=1}^{e_i-1} l_{i,j}p_i^j + sp_i^{e_i}) \mod n \\
&\equiv n(\sum_{j=1}^{e_i-1} l_{i,j}p_i^{j-1} + sp_i^{e_i-1}) \mod n \\
&\equiv 0 \mod n
\end{aligned}$$

And, since $\beta^{n/p_i} = \alpha^{nx/p_i}$, this means that

$$\beta^{\frac{n}{p_i}} \equiv \alpha^{\frac{nl_{i,0}}{p_i}} \mod p_i.$$

In the case where p_i is a small prime factor of n, $l_{i,0}$ can easily be worked out, for example by using Pollard's Rho algorithm and so can all the other $l_{i,j}$. In turn, it yields the decomposition of x_i to the base p_i. By repeating this process for all the p_i, this leads to a system of r modular equations, which can be solved using the CRT.

This strategy is expressed, from a more formal viewpoint in algorithm 6.2, which has complexity $O(\sum_{i=1}^{t} e_i(\log n + \sqrt{p_i}))$. From this complexity we clearly understand that unless n is smooth, the Pohlig Hellman algorithm will not perform well as in the worst case, i.e. when n is prime, it has complexity $\sqrt{n}$.

6.1.3 Index calculus algorithms

The index calculus method designates a way to calculate the index, as called in the 18th century, of an integer modulo a prime p, relative to a primitive root. The index, or discrete logarithm as it is now called, is best computed using this method which

Algorithm 6.2 Pohlig Hellman algorithm.

INPUT: α a generator of $\mathbb{G}$ a group of order n, and $\beta \in \mathbb{G}$.
OUTPUT: $\log_\alpha \beta$.

1: Decompose n into prime factors: $n = \prod_{i=1}^{t} p_i^{e_i}, \quad e_i \geq 1$
2: **for** $i \leftarrow 1$ to t **do**
3: $\quad l_{-1} \leftarrow 0$
4: $\quad \gamma \leftarrow 1$
5: $\quad \overline{\alpha} \leftarrow \alpha^{n/p_i}$
6: $\quad$ **for** $j \leftarrow 0$ to $e_i - 1$ **do**
7: $\quad\quad \gamma \leftarrow \gamma \alpha^{l_{j-1} p_i^{j-1}}$
8: $\quad\quad \overline{\beta} \leftarrow (\beta \gamma^{-1})^{n/p_i^{j+1}}$
9: $\quad\quad l_j \leftarrow \log_{\overline{\alpha}} \overline{\beta}$
10: $\quad$ **end for**
11: $\quad x_i \leftarrow \sum_{j=0}^{e_i - 1} l_j p_i^j$
12: **end for**
13: solve the system:

$$\begin{cases} x_1 & \equiv & x & \bmod p_1^{e_1} \\ \vdots & & \vdots & \\ x_i & \equiv & x & \bmod p_i^{e_i} \\ \vdots & & \vdots & \\ x_t & \equiv & x & \bmod p_t^{e_t} \end{cases}$$

14: **return** x

features a few different variants, all split into three phases. Note that this method only applies to fields of the form $\mathbb{F}_{p^n}$, for p a prime and n a positive integer.

Given a group $\mathbb{G} \subset \mathbb{F}_{q^k}$ of order n, the first stage consists in taking two isomorphic representations of $\mathbb{F}_{q^k}$ and fixing a subset of elements in each of the representations. Such a subset is called a *factor base* and is usually made of prime elements with norm less than a fixed bound $\mathcal{B}$. Then, the elements of the field are sieved in such a way that only the *smooth* ones, i.e. the ones completely factorising over the factor base, are kept.

Let F_1 and F_2 be two isomorphic representations of $\mathbb{F}_{q^k}$, $\alpha_1 \in F_1$ and $\alpha_2 \in F_2$. If both α_1 and α_2 are smooth on their respective factor base F_1 and F_2 and $\alpha_1 \cong \alpha_2$,

then by definition, we get the following equality:

$$\alpha_1 = \prod_{\gamma_i \in \mathcal{F}_1} \gamma_i^{a_{1,i}} \cong \prod_{\gamma_j \in \mathcal{F}_2} \gamma_j^{a_{2,j}} = \alpha_2.$$

Once sufficiently many such relations have been collected, the logarithm of these equations is taken, thus resulting in linear equations in the unknowns, the logarithms of the elements of the factor bases. The second stage consists in solving this system of equations in order to find the logarithm of the factor base elements.

Although the matrix of equations is originally sparse, after only a few operations it becomes congested, rendering the solving of the linear system a non-trivial task, that can only be achieved by using structured Gaussian elimination or more advanced algorithms, like Lanczos or Wiedemann algorithms [80, 139]. It is interesting to note that some derived algorithms can combine a few of these methods [79].

The last phase of an index calculus algorithm is computing the discrete logarithm of arbitrary elements in $\mathbb{G}$. This is achieved in ways varying with each individual algorithm, usually using a variation of the *special-*$\mathfrak{q}$ *descent*, which is defined as follows. For some element $\mathfrak{q}$ not in the factor base, sieve elements as in the first stage, searching for an element $\mathfrak{d}$, such that $\mathfrak{q}$ divides the ideal generated by $\mathfrak{d}$, and the norm of $\mathfrak{d}$ factors into primes smaller than some value $\mathcal{D}$. Then, factor the ideal generated by $\mathfrak{d}$ into a product of ideals and repeat the process until the bound $\mathcal{D}$ becomes smaller than $\mathcal{B}$, meaning that all the factors are in the factor base. Finally, $\mathfrak{q}$ is written as a product of elements in the factor base, allowing us to easily find its logarithm.

All the difficulty of this method lies in the balance of the two first stages, which are time consuming. In fact, if too many relations are collected, then the linear system

becomes too huge to be solved in reasonable time, but on another hand, if it is constituted of too few relations, then it cannot be solved. A trade off must then be found in order to have both phases to take roughly the same amount of time, the third stage being negligible compared to them.

From a general point of view, the index calculus method has a sub-exponential complexity, i.e. neither polynomial nor exponential, but "in between". More formally a sub-exponential complexity is expressed by the so called *L-notation:*

$$L_q(\alpha, c) = \exp\left((c + o(1))(\log q)^{\alpha}(\log\log q)^{1-\alpha}\right),$$

where c is a positive constant, and $0 \leq \alpha \leq 1$. We note that, $\alpha = 0$ leads to a polynomial complexity, while $\alpha = 1$, implies it to be fully exponential. When c is unknown, the L-notation is simply denoted $L_q(\alpha)$.

Before studying in more details two index calculus algorithms, we point out that factorising $q^k - 1$ can reduce the problem of finding discrete logarithms.

- For each small prime factor s of $q^k - 1$, the discrete logarithm modulo s can be computed using Pollard's Rho method.
- For the larger prime factors l of $q^k - 1$, the index calculus method can be used to compute the discrete logarithm modulo l.

Although these results can be combined to compute the logarithm modulo $q^k - 1$ using the CRT, this strategy also raise the question of knowing how to efficiently factorise $q^k - 1$. In fact, it is fairly easy if one consider the factorisation of

$$x^k - 1 = \prod_{l|k} \Phi_l(x),$$

and then substitute q for x. It is also interesting to note that even if the use of "heavy" algorithms like the number field sieve is required, this will not influence the

overall complexity of the method, as they feature the same complexity, $L_q(1/3)$, as the algorithms mentioned below.

6.1.3.1 Function field sieve algorithm

The Function Field Sieve (FFS), due to Adleman [4], is an algorithm based on the index calculus method which has the particularity of targeting fields of small characteristic. As such it is especially suitable in the case of supersingular pairing-friendly elliptic curves, defined over fields of characteristic 2 and 3, with maximal embedding degrees $k = 4$ and $k = 6$ respectively. Thus, in this context, the pairings will map the ECDLP over $\mathbb{F}_{2^m}$ to the DLP over $\mathbb{F}_{2^{4m}}$ and the ECDLP over $\mathbb{F}_{3^m}$ to the DLP over $\mathbb{F}_{3^{6m}}$, for m a prime, or a near prime.

The FFS, as for all index calculus algorithms, follows the main pattern mentioned above, but has the special property of representing the elements of the field as polynomial functions. Although a few variants exist, we will focus on the most efficient version to date, which is due to Joux and Lercier [73].

Before going further into the details of the algorithm, we first define the field $\mathbb{F}_{p^{km}}$ using an irreducible polynomial $f(x)$ over $\mathbb{F}_p[x]$, and set a constant

$$d = \left\lceil \sqrt{\frac{km}{(\frac{4}{9})^{1/3}(km)^{1/3}\log_p(km)^{2/3}}} \right\rceil .$$

In the original article presenting the FFS to solve the DLP, Adleman uses a general bi-variate polynomial that must satisfy a list of eight conditions. However, Joux and Lercier realised that, using a special class of bi-variate polynomials, called C_{ab} *curves*, only two of the eight original conditions needed to be satisfied. Reminding that a *perfect field* is a field where every algebraic extension is separable, the following proposition gives a definition of such C_{ab} curves.

Proposition 6.1.1 ([87]). *Let* $\mathbf{K}$ *be a perfect field,* $\bar{\mathbf{K}}$ *the algebraic closure of* $\mathbf{K}$*,* $\chi \subset \bar{\mathbf{K}}^2$ *be a possibly reducible, affine algebraic set defined over* $\mathbf{K}$*,* x, y *be the coordinates of the affine space and* a, b *relatively prime positive integers. Then the following conditions are equivalent:*

- χ *is an absolutely irreducible affine algebraic curve with exactly one* $\mathbf{K}$*-rational place* P *at infinity and the pole divisors of* x *and* y *are* aQ *and* bQ *respectively.*
- χ *is defined by a bi-variate polynomial of form*

$$H(x, y) = \alpha_{b,0}x^b + \alpha_{0,a}y^a + \sum_{ia+jb<ab} \alpha_{i,j}x^i y^j$$

  where $\alpha_{i,j} \in \mathbf{K}$ *for all* i, j *and* $\alpha_{b,0}$*,* $\alpha_{0,a}$ *are nonzero, such a curve is called a* C_{ab} *curve.*

The two remaining conditions, in order to be able to use $H(x, y)$ in the FFS, are stated as follows:

- $H(x, \chi(x))$ is divisible by f, where $\chi(x)$ is some random polynomial of degree at most $\lfloor km/d \rfloor$.
- The order h of the Jacobian of the curve defined by $H(x, y)$ is prime to $(p^{km} - 1)/(p - 1)$.

These being set, we follow the strategy adopted by Joux and Lercier in [73]. At first only p^{km} is fixed, then a $C_{a,b}$ curve, defined by a bi-variate polynomial $H(x, y)$ of degree d is chosen. Note that no irreducible polynomial $f(x)$ has been chosen at this stage. To construct it, two polynomials $\chi_1(x)$ and $\chi_2(x)$ of degree at most $\lfloor km/d \rfloor$ are picked randomly. If the polynomial defined by

$$\chi_2(x)^d H(x, -\chi_1(x)/\chi_2(x))$$

is irreducible and has degree km, then it defines the polynomial $f(x)$. Otherwise, a new pair of polynomials $(\chi_1(x), \chi_2(x))$, is selected until a suitable pair of polynomials has been found.

When choosing $H(x, y)$ and generating χ_1 and χ_2, two cases must be considered:

- if $d|km$, then $\deg_x(\chi_1) = \frac{km}{d}$ and $d.\deg_x(\chi_2) + \deg_x(H(x, y)) < km$
- if $d \not| k$, then $d.\deg_x(\chi_2) + \deg_x(H(x, y)) = km$

Once $H(x, y)$, $\chi_1(x)$, $\chi_2(x)$ and $f(x)$ have been properly constructed the following homomorphism can then be defined:

$$\phi : \begin{array}{ccc} F_2 = \mathbf{K}[x, y]/(H(x, y)) & \longrightarrow & F_1 = \mathbf{K}[x]/(f(x)) \\ y & \longmapsto & -\frac{\chi_1}{\chi_2} \end{array}$$

As described in section 6.1.3, the goal of the function field sieve is to find doubly smooth elements. Therefore we define the first factor base, called the algebraic factor base $\mathcal{F}_2$, as being composed of small prime divisors in the divisor group $\mathrm{Div}(F_2)$, while the second, called the rational factor base $\mathcal{F}_1$, consists of small degree, irreducible polynomials in $\mathbf{K}[x]$. Thus, $\mathcal{F}_1$ and $\mathcal{F}_2$ are defined by

$$\mathcal{F}_2 = \{\langle p(x), y - \tau\rangle |\ \deg p(x) \leq \mathcal{B}, p(x) \text{ irreducible and } \tau \equiv -\chi_1/\chi_2 \bmod p(x)\}$$

$$\mathcal{F}_1 = \{p(x) |\ \deg p(x) \leq \mathcal{B} \text{ and } p(x) \text{ irreducible}\}.$$

Let $r(x)$ and $s(x)$ be two coprime polynomials of degree $\mathcal{B} \leq (\frac{4}{9}km)^{1/3} log_p(km)^{2/3}$. Suppose that the divisor $\langle s + r.y\rangle$ can be factorised over $\mathcal{F}_2$ into a product of small prime divisors $\langle p_i(x), y - \tau_i\rangle$. Recalling that we define h as the order of the Jacobian of a $C_{a,b}$ curve, each $h\langle p_i(x), y - \tau_i\rangle$ is a principal divisor, and as ϕ is an homomorphism

the following algebraic relation holds

$$(s + r.y)^h = u \prod_{\lambda_i} \lambda_i^{a_i},$$

where u is in $\mathbf{K}^*$, λ_i are uniquely defined functions in F_2 and a_i are positive integers. Applying ϕ results in an equality modulo a factor in $\mathbf{K}^*$

$$\left(s - r.\frac{\chi_1}{\chi_2}\right)^h = \prod_{\lambda_i} \phi(\lambda_i)^{a_i}.$$

As the above second condition must hold, h is coprime to $(p^{km} - 1)/(p - 1)$, and thus, the hth roots can be taken on both sides of the equation, yielding

$$\left(s - r.\frac{\chi_1}{\chi_2}\right) \equiv \prod_{\lambda_i} \nu_i^{a_i},$$

where $\nu_i = \phi(\lambda_i)^{1/h}$. Hence, if the polynomial $s\chi_2 - r\chi_1$ factorizes over $\mathcal{F}_1$ into small degree irreducible polynomials $p_j(x)$, then the equation can be rewritten

$$\frac{1}{\chi_2} \prod_{p_j(x)} p(x)_j^{b_j} \equiv \prod_{\lambda_i} \nu^{a_i}.$$

The following relation between discrete logarithms is then obtained

$$\sum_{p_i(x)} b_j \log p_j(x) - \log(\chi_2) = \sum_{\lambda_i} a_i \log \nu_i,$$

Once sufficiently many independent relations have been found, their discrete logarithms, can be worked out using linear algebra techniques as briefly explained in section 6.1.3.

It is also interesting to note that, according to the work of Granger, Holt, Page, Smart and Vercauteren [55], this technique can be adapted to use a superelliptic curve instead of a normal $C_{a,b}$ curve, to compute the discrete logarithm problem in a finite

field of characteristic three. Using superelliptic curves makes algebraic computation more efficient, but at the same time introduces three different cases, depending on the $\gcd(km, d)$ being 1, d or another value. In fact they noted that, if three divides d, then superelliptic curves cannot be used, and should instead be replaced by a $C_{a,b}$ curve.

Once the discrete logarithm of the elements of the factor base is known, the next stage is to compute the discrete logarithm of any given element $e(t)$. In order to reach this end, we follow the method proposed by Joux and Lercier [73].

Using an element of the factor base $b(t)$, we generate a new polynomial $a(t) = b(t)^i e(t)$, where i is a positive integer. As $b(t)$ is in the factor base, the discrete logarithm of $e(t)$ can easily be recovered from $a(t)$, and then, using the extended Euclidean algorithm, $a(t)$ can be written $a_1(t)/a_2(t)$ with $a_1(t)$ and $a_2(t)$ two polynomials of degree around $km/2$. If $a_1(t)$ or $a_2(t)$ is not smooth with respect to the bound $L_{p^{km}}(2/3)$, a special-$\mathfrak{q}$ descent is used to compute their discrete logarithm. In turn, this strategy leads to the discrete logarithm of $a(t)$.

When the optimal value d defined above is used, the FFS results in a subexponential algorithm of complexity

$$L_{p^{km}}(1/3, (32/9)^{1/3}) = \exp\left(\left(\left(\frac{32}{9}\right)^{\frac{1}{3}} + o(1)\right)\log(p^{km})^{\frac{1}{3}}\log(\log(p^{km}))^{\frac{2}{3}}\right), \quad p = 2, 3.$$

6.1.3.2 Number field sieve algorithm

The FFS targets fields of small characteristic. It cannot be used to solve the DLP in fields resulting from pairings over ordinary pairing-friendly elliptic curves. However in this new context, another variant of the index calculus method, called the Number Field Sieve (NFS), can be used.

As for any algorithm based on the index calculus method it features the three usual steps: sieving, solving a linear system of equations, and extending the computation

of discrete logarithms from the factor base to all the elements in the field. However, instead of considering elements as functions as in the FFS case, they are now viewed as numbers. To analyse this new approach we follow the improvements introduced by Joux, Lercier, Smart and Vercauteren [75], to the original NFS algorithm [119].

The computation is performed in fields of the form $\mathbf{K} = \mathbb{Q}[\theta]$, for θ a root of an irreducible polynomial over $\mathbb{Z}[x]$, denoted $f(x)$. At this stage it is important to note that, although the ring of integers of $\mathbf{K}$, $\mathfrak{O}_{\mathbf{K}}$, may not be a unique factorisation domain it still is a Dedekind domain, implying the existence of a unique factorisation over ideals. Therefore, the NFS will not deal with the numbers themselves but rather, with the ideals they generate, one of the aims being their factorisation.

Not forgetting that the primary goal is to give two isomorphic representations of $\mathbb{F}_{p^k}$, we notice that, if f has degree k and p remains inert in $\mathfrak{O}_{\mathbf{K}}$, then $\mathfrak{O}_{\mathbf{K}}/(p)$ is a field isomorphic to $\mathbb{F}_{p^k}$. In some cases it may happen that f cannot be taken of degree exactly k, but only of degree $l > k$. To overcome this issue we recall that prime ideals can be factored over $\mathfrak{O}_{\mathbf{K}}$ and as such, if (p) is not inert but splits, i.e.

$$(p) = \prod_i \mathfrak{p}_i^{e_i},$$

and is unramified in $\mathfrak{p}_j$, with inertia degree $f_j = k$, then it is sufficient to consider the residue field $\mathbb{F}_{p^k} \cong \mathfrak{O}_{\mathbf{K}}/\mathfrak{p}_{\mathrm{j}}$. We denote by $\psi_{\mathcal{I}}$, the map from $\mathfrak{O}_{\mathbf{K}}$ to $\mathbb{F}_{p^k}$, and $\overline{x}$ the image of x.

In the basic variation of the NFS, when $p \approx L_{p^k}(2/3, 2/3^{1/3})$, the sieving occurs over elements of the form $a - b\theta$. However as in the context of pairings the prime p is smaller and the embedding degree k is larger, the sieving space must be extended in order to collect enough relations to be able to solve the final system of equations.

Therefore, the sieving must go through elements written in the more general form

$$\sum_{i=0}^{l} a_i \theta^i, \quad l \leq k.$$

The first step of the algorithm, consists in picking a polynomial $f_1(x)$ which is irreducible over $\mathbb{F}_p[x]$, has degree k and preferably has small coefficients. Then, defining $f_2(x)$ to be $f_1(x) + p$, it is clear that $f_1(x)$ and $f_2(x)$ have all the same roots modulo p as they are equal modulo p and as such, have a common root in $\mathbb{F}_{p^k}$.

We can then define F_1 and F_2, to be two algebraic number fields such that, $F_1 = \mathbb{Q}[\theta_1]$ and $F_2 = \mathbb{Q}[\theta_2]$, for θ_1 and θ_2 zeros of f_1 and f_2 in $\mathbb{C}$, respectively. This setup is represented in diagram 6.1.

$$\begin{tikzcd}
 & \mathbb{Q}[x] \arrow[dl, dash] \arrow[dr, dash] & \\
F_1 = \mathbb{Q}[x]/f_1 \arrow[d, dash] & & \mathbb{Q}[x]/f_2 = F_2 \arrow[d, dash] \\
\mathcal{O}_{F_1}/(p) = \mathbb{F}_{p^k} \arrow[rr, dash] & & \mathbb{F}_{p^k} = \mathcal{O}_{F_2}/(p)
\end{tikzcd}$$

Figure 6.1: Diagram showing the setup of the NFS over $\mathbb{F}_{p^k}$.

Once this fields are defined, the next stage consists in constructing the factor bases. This is achieved using the following lemma [75].

Lemma 6.1.2. *Let* $\mathbf{K} = \mathbb{Q}[\theta]$ *and* $(a_0, \ldots, a_l)$ *be an* $(l+1)$*-tuple of integers, with* $\gcd(a_0, \ldots, a_l) = 1$*, then a prime ideal* $\mathfrak{p}$ *dividing the principal ideal generated by* $\sum_{i=0}^{l} a_i \theta^i$*, either has norm dividing* $f_\theta = [\mathfrak{O}_{\mathbf{K}} : \mathbb{Z}[\theta]]$ *or has degree* $\leq l$*.*

This allows us to set up the factor bases to be the set of prime ideals which either,

have degree less than l, or divide the index f_θ.

$$\mathcal{F}_i = \{\text{prime ideals } \mathfrak{p} \text{ of norm } < \mathcal{B}_i \ / \ \deg \mathfrak{p} \leq l \text{ or } N_{F_i/\mathbb{Q}}(\mathfrak{p}) | f_{\theta_i}\},$$

for some l that will be determined later, and $f_{\theta_i} = [\mathfrak{O}_{F_i} : \mathbb{Z}[\theta_i]]$, $i = 1, 2$. From the above factor bases set up, it is clear that appropriate values for the smoothness and sieving bounds, $\mathcal{B}_1$, $\mathcal{B}_2$ and $\mathcal{S}$ must be determined at some stage.

While sieving, elements represented as $(l+1)$-tuples $(a_0, \ldots, a_l)$, $a_i \in \mathbb{Z}$ and satisfying the following properties are sought:

- $gcd(a_0, \ldots, a_l) = 1$,
- $|a_i| \leq \mathcal{S}$
- $\sum_{i=0}^{l} a_i \theta_j^i$, for $j = 1, 2$, have $\mathcal{B}$-smooth norms in F_1 and F_2 respectively.

The norms of these $(l+1)$-tuples in F_1 and F_2 are given by the resultants of the polynomials $A(x) = \sum_{i=0}^{l} a_i x^i$ with $f_1(x)$ and $f_2(x)$ respectively, that is,

$$N_{F_j/\mathbb{Q}}\left(\sum_{i=0}^{l} a_i \theta_j^i\right) = Res(A(x), f_j(x)), \quad j = 1, 2.$$

Once sufficiently many smooth elements have been found, the ideal generated by $\sum_{i=0}^{l} a_i \theta_j^i$, $(j = 1, 2)$, is factored into a product of ideals from the respective factor bases $\mathcal{F}_j$, $(j = 1, 2)$. As the norm of these elements has already been determined to be $\mathcal{B}_j$-smooth, they can be written

$$N_{F_j/\mathbb{Q}}\left(\sum_{i=0}^{l} a_i \theta_j^i\right) = \prod_t p_t^{e_t}, \quad p_t \text{ prime s.t. } p_t \leq \mathcal{B}_j, j = 1, 2. \tag{6.1.2}$$

At this point, a bit more work is required in order to give the prime ideal factorisation of $(A(\theta))$. Therefore, we introduce the following result, that can be found in [100, chapter 1, proposition 8.3].

Proposition 6.1.3. *Let $\mathfrak{p}$ be a prime ideal which does not divide the ideal $\mathfrak{F} = \{\alpha \in \mathfrak{O} \ / \ \alpha\mathfrak{O}_{\mathbf{K}} \subseteq \mathfrak{O}_{\mathbf{K}}[\theta]\}$, and $f(x) = \prod_i g_i(x)^{e_i} \mod \mathfrak{p}$, the factorisation of $f(x)$, the minimal polynomial of θ, modulo $\mathfrak{p}$ over the residue class field $\mathfrak{O}_{\mathbf{K}}/\mathfrak{p}$. Then,*

$$\mathfrak{p}_{\mathrm{i}} = \mathfrak{p}\mathfrak{O}_{\mathbf{K}} + g_i(\theta)\mathfrak{O}_{\mathbf{K}},$$

where the $\mathfrak{p}_i$ are some prime ideals above $\mathfrak{p}$. Moreover, each $\mathfrak{p}_i$ has inertia degree f_i equals to the degree of $g_i(x)$, and

$$\mathfrak{p} = \prod_i \mathfrak{p}_i^{e_i}.$$

Unfortunately, this result only helps to find the prime ideals occurring in the factorisation of $(A(\theta))$, without giving any information on their valuations. Therefore we recall a result stating that if $\mathfrak{p}$ is a prime ideal in $\mathfrak{O}_{\mathbf{K}}$, then there exists $a \in \mathbf{K}\backslash\mathfrak{O}_{\mathbf{K}}$ such that $a\mathfrak{p} \subset \mathfrak{O}_{\mathbf{K}}$, and if $\mathfrak{p}$ divides a given ideal I, then the ramification index of I in $\mathfrak{p}$ is the largest v such that $a^v I \subset \mathfrak{O}_{\mathbf{K}}$ [26, section 4.8.3].

Once the practical principles underlying the prime factorisation of ideals have been given, they can be applied to our case of concern, $(A(x))$. For each p_t in equation 6.1.2, not dividing the index f_θ, (p_t) can be factorised into a product of prime ideals

$$(p_t) = \prod_i \mathfrak{p}_{t_i}^{e_{t,i}}.$$

Then, each irreducible factor of $\gcd(A(x), f(X))$ over $\mathbb{F}_{p_t}$, leads to an ideal $\mathfrak{p}_{t_i}$ lying above p_t. To find its valuation it suffices for us either to calculate $k/\deg \mathfrak{p}_{t_i}$ if the gcd is irreducible, or apply algorithm 4.8.17 from [26]. It is interesting to note that this algorithm also applies to the case where $p_t | f_\theta$.

As the factorisation into prime ideals has been achieved it means that some relations

over the ideals have been found. However, in order to find the discrete logarithm of elements in $\mathbb{F}_{q^k}$, we need the relations to be over the numbers. Therefore, we now discuss the conversion of relations over ideals into relations over the field elements. At this stage two situations may arise, depending on the class number of $\mathbf{K}$, and the existence of a computable unit group.

We start by studying the easiest case, that is, when the class number of $\mathbf{K}$ is 1, implying that $\mathfrak{O}_{\mathbf{K}}$ is a principal domain. Thus, for all ideals $\mathfrak{p}_j = (p_j, \theta - c_{p_j})$ there exists an element $\gamma_j \in \mathbf{K}$ such that $\mathfrak{p}_j = (\gamma_j)$. In this case, any ideal $(a - b\theta)$ can be written as a product of these elements: $(a - b\theta) = u \prod_j \gamma_j^{e_j}$ where u is a unit in $\mathfrak{O}_{\mathbf{K}}$. Let (r_1, r_2) be the signature of $\mathbf{K}$ and define $r = r_1 + r_2 - 1$, then the unit group of $\mathbf{K}$ is denoted $\mathcal{U}_{\mathbf{K}} = \mathfrak{O}_{\mathbf{K}}^* \cong \nu(\mathbf{K}) \times \mathbb{Z}^r$, where $\nu(\mathbf{K})$ is a finite cyclic group of order ω, $\nu(\mathbf{K}) = \langle u_0 \rangle$. By assuming that $\mathbf{K}$ has a computable unit group, it becomes possible to compute the r fundamental units: $u_1, \ldots, u_r$, and u can be written as a product of these fundamental units and the generator of $\nu(\mathbf{K})$, $u = u_0^{n_0} \ldots u_r^{n_r}$.

Using this decomposition into a product of fundamental units, r logarithmic maps Λ_i, for $i = 1, 2, \ldots, r$ are defined:

$$\begin{aligned} \Lambda_i : \mathcal{U}_{\mathbf{K}} &\rightarrow \mathbb{Z} \\ u &\mapsto n_i. \end{aligned}$$

Similarly, the logarithmic map Λ_0 is defined:

$$\begin{aligned} \Lambda_0 : \mathcal{U}_{\mathbf{K}} &\rightarrow \mathbb{Z} \\ u &\mapsto n_0. \end{aligned}$$

Thus, the final decomposition of $a - b\theta$ is given by:

$$a - b\theta = \prod_{i=0}^{r} u_i^{\Lambda_i(u)} \prod_i \gamma_i^{e_i}.$$

The next step is then to apply the map $\psi_{\mathcal{I}}$ to each side of the equation and take the logarithms, resulting in:

$$\log(a - b\theta) = \sum_{i=1}^{r} \Lambda_i(u) \log \bar{u}_i + \sum_i e_i \log \bar{\gamma}_i \bmod p^n - 1.$$

Note that $\log \bar{u}_i$, and $\log \bar{\gamma}_i$ introduce new unknowns to the system of equations, implying that at least $r + |\mathcal{F}|$ linear independent equations need to be found, in order to be able to solve the final linear system.

When dealing with the general case, one must be careful as the large subgroup in which the computation is performed should have order not dividing the class number of $\mathbf{K}$ denoted h. In fact, this is only a minor restriction as remarked in [75].

We recall that the ideal decomposition has already been obtained as

$$(a - b\theta) = \prod_j \mathfrak{p}_j^{e_j}.$$

Then, raising both sides of the equation to the power h, yields

$$(a - b\theta)^h = u \prod_j \delta_j^{e_j}, \quad \delta_j^{e_j} \in \mathcal{O}_{\mathbf{K}} \text{ s.t. } (\delta_j) = \mathfrak{p}_j^h.$$

At this stage, a second difficulty is encountered as the most straight forward idea would be to take the logarithms on both sides. However since there is no assumption of a computable unit group, a basis of fundamental units for the factorisation of u can not necessarily be computed. Each equation is likely to have a different u, so taking the logarithms at this stage would introduce too many new unknowns and thus drastically increase the number of equations needed and the time to solve for the unknowns. The solution to overcome this issue relies on the fact that as the logarithms are being computed modulo l it is sufficient to work with the unit group modulo the

lth powers of units, for if $u \in (\mathcal{U}_{\mathbf{K}})^l$, then $\log \bar{u} \equiv 0 \bmod l$. Joux, Lercier, Smart and Vercauteren showed that the problem can be completely solved by using an adaptation of the Schirokauer algorithm [118]. However, this being mostly a technical issue we refer the reader to [75] for more details on the topic.

Assuming that the logarithms of the elements in the factor bases have been calculated, the next step is to calculate the discrete logarithm of any arbitrary element.

Following the method used in [75] for finding the discrete logarithm of an individual element x, we start by representing $\mathbb{F}_{p^k}$ as the field $\mathbb{F}_p[t]/(f_1(t))$ and find an element $y \in \mathbb{F}_p[t]/(f_1(t))$, $y = x^i t^j$ for some non negative integers i and j such that:

- $y \in F_1$ is $\mathcal{B}_1$ smooth,
- The factorisation of $N_{F_1/\mathbb{Q}}(x)$ contains prime powers $\leq l$.

After finding such an element y, the principal ideal generated by y should factor over $\mathcal{F}_1$. If there are some factors of (y) not in $\mathcal{F}_1$, then the logarithms of these elements are computed using *special*-$\mathfrak{q}$ descent as described in section 6.1.3.

The above investigation points out the complexity and flexibility of the NFS algorithm, which lead to different variations with several subcases. The main drawback is that no fixed complexity can be expressed. However it is obvious that it depends on the size of the sieving space, given by the bounds $\mathcal{S}$ and $\mathcal{B}$, and the degree l of the elements over which the sieving is performed. Again, the value for l varies depending on the ratio of the values of p and k, being given by the real zero of the polynomial $3c^3 l(l+1)^2 - 32 = 0$, where

$$c = \frac{1}{k}\left(\frac{\log q}{\log\log q}\right)^{1/3} = \log p(\log^2 q \log\log q)^{-1/3}.$$

Given that l can vary, so does the size of the sieving space and hence the com-

plexity of the whole algorithm. The bounds are given by $\mathcal{S} = L_{p^k}(1/3, c')$ and $\mathcal{B} = L_{p^k}(1/3, 2c'/(l+1))$, where

$$c' = \frac{1}{3}\left(\frac{2}{(l+1)c} + \sqrt{\frac{4}{(l+1)^2c^2} + 3lc}\right).$$

For this given size of sieving space and taking into account the probability of smoothness [75] the overall complexity of the algorithm is given by $L_{p^k}(1/3, 2c')$.

6.2 Practical view

One of the main issues faced by the theoretical view is that it leads to asymptotic complexities, not necessarily reflecting the real world. Therefore, having efficient implementations of the different algorithms presented in the previous section would be of a great help in our investigation.

Regarding Pollard's Rho algorithm, there exists a lot of small improvements [133], including parallelized implementations [137], however it is clear that its complexity will remain exponential. Knowing the exact cost of the ECDLP is of a major importance in the context of pairings as for families of curves having a ρ-value close to 1 the DLP, solved using sub-exponential algorithms, must cost the same as solving the ECDLP using Pollard's Rho algorithm.

As for Pohlig Hellman algorithm, it is very efficient in certain rare cases that are easily avoided, by picking a prime p such that the order of the large subgroup, in which the DLP lies, is not smooth. Hence, implementing it in order to break the DLP in general cases would result in a totally inefficient solution. Therefore, we now focus on the two algorithms based on the index calculus method, namely the FFS and the NFS.

Note that Magma [3] does not implement either of these two algorithms. Instead a version of the Coppersmith algorithm [27] by Thome [135] is used in characteristic 2, while for extension fields of characteristic $p > 2$ the Pohlig Hellman algorithm is the

best available.

6.2.1 FFS algorithm

Although really few implementations of the FFS exist, it is interesting to note that the last to date exactly targets the context of pairings. More precisely, Hayashi, Shinohara, Wang, Matsuo, Shirase, and Takagi implemented the FFS over fields of the form $\mathbb{F}_{3^{6m}}$ and were able to solve the DLP over $\mathbb{F}_{3^{6.71}}$ [65].

More than a simple implementation of the FFS, their article compares two versions of the algorithm, both due to Joux and Lercier. The first variant is the one presented above, targeting primarily fields of characteristic 2 or 3, while the second one is supposed to fit fields of medium characteristic [74]. Surprisingly, the second one appears to perform better in practice, featuring a more efficient sieving stage, based on a polynomial sieve instead of a lattice sieve. Also note that this second variation of the FFS has a worse asymptotic complexity.

The particularity of this version of the FFS is that it has a noticeably smaller sieving space, still containing enough smooth elements, hence largely improving the probability for an element to be smooth. Recalling that the goal is to find pairs of polynomials $(r(x), s(x))$ which are doubly smooth, and that the homomorphism ϕ maps an element $y \in \mathbf{K}[x, y]/(H(x, y))$ to $M \in \mathbf{K}(x)/(f(x))$, they fixed $s(x)$ and tried to find $r(x)$ such that $r(x)M + s(x)$ is divisible by an irreducible polynomial $p(x)$ belonging to the factor base. Then, remarking that $r(x)M + s(x) + k(x)p(x)$, $k(x) \in \mathbb{F}_{p^k}[x]$, is also divisible by $p(x)$, it is fairly easy to obtain all the polynomials $r(x)$ of degree less than a bound $\mathcal{B}$ such that $r(x)M + s(x)$ is divisible by $p(x)$. Therefore, when all $r(x)$ have been computed for each $p(x)$, it suffices to check whether or not the degree of $r(x)M + s(x)$ is equal to the sum of the degrees of all the $p(x)$ dividing it. If so, considering $\gcd(r(x), s(x))$ will lead to a new relation as soon as it is equal to 1.

Together with this sieving technique they used a parallelized implementation of the Lanczos method to solve the linear algebra part. By the end they were able to solve the DLP in $\mathbb{F}_{3^{426}}$, a field containing a subgroup of order a 112 bit prime. The computation was completed within 33 days, using a cluster with four nodes, each consisting of Intel Quad-Core Xeon E5440 (2.83 GHz) 2 CPUs with 16-GB RAM, and three clusters with four nodes, each consisting of Intel Quad-Core Xeon L5420 (2.33 GHz) 1 CPU with 4-GB RAM.

This new record is quite interesting as it is the first attempt at targeting supersingular curves over ternary fields. Until now mostly fields of characteristic 2 were studied, the last record to date being the resolution of the DLP by Joux [71] over $\mathbb{F}_{2^{613}}$. Once the case of supersingular elliptic curves has been examined in practice the next stage consists in solving the DLP in the context of ordinary pairing-friendly elliptic curves, using the NFS algorithm.

6.2.2 NFS algorithm

Implementing the NFS is not an easy task, in fact there is no record of any general implementation. According to the authors, the code used in [75] targets some specific "easy" cases under which the DLP in the context of pairings does not fall. Thus, there was no other choice than implementing the NFS from scratch.

In order to run some toy examples, we decided to start by implementing a linear sieve, whose code, based on the GMP [2] and NTL [126] libraries, is given in appendix B. Once it was running accurately and efficiently, it was possible to test our fast Gaussian matrix reduction implemented using the MIRACL library [121].

One of the main issues, when implementing the NFS, is the hardness of some underlying problems. The first one to mention, is the determination of the class number. As

explained in the previous section, the main idea of the index calculus method is to define two isomorphic representations of a given finite field, which from a theoretical point of view is easy to achieve. However, when looking at the construction of the polynomials $f_1(x)$ and $f_2(x)$, one realises that although $f_1(x)$ is sparse and has small coefficients $f_2(x) = f_1(x) + p$ has a very large coefficient in its degree 0 monomial. In turn, the presence of this large coefficient renders the class number of the field $\mathbb{Q}[x]/f_2(x)$ hard to compute [134]. Note that the closely related question of determining the unit group is also hard to solve [20, 5], although certain cases in the NFS algorithm assume it is known.

At this stage one of the most important things to realise is that even though these parameters are required by the algorithm and are hard to figure out, they can be precomputed and as such, will not influence the overall time needed to complete the calculation of the discrete logarithm of a given element. Therefore one should focus on the sieving stage.

In the context of pairings, the sieving must occur over a higher dimensional space in order to get enough smooth relations to solve the linear system of equations. The main problem arising here is that increasing the dimension of the sieving space results in a large increase of the size of the norm of the ideals. Recalling that the main goal of the sieving stage is to find smooth elements, it becomes evident that a huge amount of factorisations of very large integers will be involved. Therefore some very efficient smoothness test must be used.

The solution we adopted tries to take advantage of the best known algorithms for factorising. We first start by removing all the small prime factors under ≈ 50000 using trial division, then the Pollard's Rho algorithm for factoring allows to find more small factors, larger than 50000. Next, the elliptic curve method for factorising is used to find medium size factors, while a quadratic sieve finds factors of the remaining composite

part. Note that when a composite number is found, the Pollard's Rho algorithm and the elliptic curve method are performed recursively in order to work out its factorisation into primes. Although efficient, this approach requires some refinements in order to fit to the context of a smooth test.

In fact, if one realises that having too many large factors, i.e. factors beyond the smoothness bound, is useless, then it becomes clear that when more than one "too large" factor is found in a relation it does not give any useful information and as such can be dropped. We decided to only keep relations including at most one large factor, expecting some collisions between large factors, potentially leading to new relations. If for a given large factor, no collision is found, then the relation in which it appears is useless and can be forgotten. Note that, although some variants allow two large primes, they in turn, result in a need for more collisions and a lot more complex implementations [135] involving the use of graphs.

In practice this implementation performs well, as on average it takes only $\approx 0.35s$, on an Intel Core2 Duo CPU @3.00GHz, using only one thread, in order to run a smoothness test on a ≈ 400 bit integer, while Magma [3] needs over a second. The factorisation and smoothness test code, taking advantage of the GMP [2], GMP-ECM [1] and FLINT [64] libraries, are given in appendix B.

The problem of fast smoothness testing being solved, the next issue to address is how to handle the sieving space. Recalling that smooth relations must be found in two different fields isomorphic to $\mathbb{F}_{q^k}$, it means that the norms of the elements have to be computed in both representations. As the norm of an element $\sum_{i=0}^{l} a_i\theta^i$ in a field $\mathbb{Q}[\theta]$, for some algebraic integer θ with minimal polynomial $f(x)$, is given by the resultant of the element considered as a polynomial in x with $f(x)$, it is clear that its size greatly depends on the size of the coefficients of $f(x)$. In the case of $f_1(x)$, it results in small norms easy to factor and which most of time, are smooth, while on the other side,

the constant terms in $f_2(x)$ renders the resultant about p^l times larger. Therefore, in practice finding doubly smooth elements is very hard, even using the usual trick consisting of unbalancing the coefficients of the elements sieved, i.e. taking the a_i of completely different sizes.

In fact there is not best way to generate the polynomials $f_1(x)$ and $f_2(x)$. At the moment the best that can be done is to balance the difficulty of the factorisation between the two sides by taking both $f_1(x)$ and $f_2(x)$ with coefficients of the same size as the square root of p. For a 55 bits MNT prime, that is a prime of the form $x^2 + 1$ with $x \in \mathbb{Z}$, this technique leads to a significant improvement as in practice the norms on both sides are of size about 190 to 210 bits, compared to the initial 400 bits on the $f_2(x)$ side.

However, in this case another problem appears as k remains fixed while p is a lot smaller than in the normal MNT setup where p is about 160 bits (discussed in the next section). This implies a larger value for l which depends on the ratio of p to k (end of section 6.1.3.2). In this context the sieving space has an extra dimension leading to results not mirroring the difficulty of the problem as it arises in practice.

Note that the technique proposed in [75] for p larger than $L_p(2/3)$, can be adapted to our case but will not give any improvement as either $f_2(x)$ is taken of degree not much larger than the degree of $f_1(x)$ and then the norm will remain larger than when $f_1(x)$ and $f_2(x)$ have coefficients of order $\sqrt{p}$ or the norm will blow up because of the degree of $f_2(x)$ being too large.

It is important to note that, although the NFS has an asymptotic subexponential complexity, it features many hard underlying problems which render it really difficult to implement, the main problem relying on the choice of the polynomials. In fact, in the set up $f_1(x)$ and $f_2(x)$ must be two irreducible polynomials of the same degree and sharing a common root in $\mathbb{C}$, which is highly restrictive. Therefore, unless a new set up or a new algorithm, is discovered, it is quite unlikely that the DLP can be solved

efficiently in the context of ordinary pairing-friendly elliptic curves at the parameters sizes of current cryptographic interest as we will now discuss.

6.3 Discussion

Among the four algorithms investigated, only two are really suitable to break the DLP or the ECDLP in the context of pairings. In fact, although Pollard's Rho algorithm benefited from a lot of improvements [133, 137] it still remains exponential, rendering it useless over the finite fields resulting from the pairings. However note that it is still useful in order to determine the hardness of the ECDLP. At first glance, the Pohlig Hellman algorithm has the advantage of being very efficient, however it is sufficient for the field $\mathbb{F}_q$ to have a subgroup of order a large prime to render it completely ineffective. In this regard, note that in the context of pairings the DLP must be solved in a large subgroup of size r, $r|\Phi_k(p)$. Therefore, the best to date are the two algorithms based on the index calculus method, namely the FFS and the NFS. Table 6.1 summaries the main characteristics of these four algorithms.

Algorithm	Problem	Complexity	Remark
Pollard's Rho	(EC)DLP	Exponential	No special structure required
Pohlig Hellamn	DLP	Polynomial	$p^k - 1$ must be smooth
FFS	DLP	Subexponential	Characteristic 2 or 3
NFS	DLP	Subexponential	Characteristic $p > 3$

Table 6.1: Algorithms to solve the DLP and the ECDLP

In order for the ECDLP to be unsolvable in the context of pairings, one should ensure the hardness of the DLP, being aware that the efficiency of the algorithms used for the ECDLP and the DLP is not necessarily equivalent. To make it more concrete, we introduce the notion of security level, based on complexity. In fact, one assumes that a problem cannot "possibly" be solved, in a reasonable amount of time and using the best available technology if it has complexity larger than a given bound. Today

the most common one is 2^{80}, but it is slowly steering toward the more secure level of 2^{128} [101]. Note that often a security level of 2^n, is said to be n bit secure in reference to the level of security provided by the Advanced Encryption Standard (AES) [83].

Hence, in order to make sure that both the ECDLP and the DLP are hard enough not to be solved, it suffices us to chose some parameters such that, the best algorithm to solve those problems have complexity, for instance, larger than 2^{80}. In this case, on the elliptic curve side the best algorithm available being the Pollard's Rho algorithm, the elliptic curve should have a subgroup such that the size of its order is approximatively 160 bits, while on the finite field side it should be of size about 1024 bits to be secure against attacks taking advantage of index calculus methods.

Recalling that a pairing maps a subgroup of an elliptic curve over $\mathbb{F}_q$, to a subgroup of $\mathbb{F}_{q^k}$, with k the embedding degree, it becomes clear that the embedding degree must be chosen carefully, in order not to map into a "weak" subgroup of the finite field. Assuming that a pairing-friendly elliptic curve with ρ-value close to 1 has been picked, the appropriate embedding degree is determined simply by dividing the size of the field by the corresponding size of the elliptic curve group, for a given security level. For example, at the 80 bit security level, $\frac{1024}{160} \approx 6$, so at this level, $k = 6$ would be an appropriate embedding degree. Table 6.2 shows the approximate equivalence of the efficiency of the current algorithms for both the ECDLP and the DLP [43].

Security level (in bits)	ECDLP group size (in bits)	DLP group size (in bits)	Embedding degree ($\rho = 1$)
80	160	960 - 1280	6-8
128	256	3000 - 5000	12-20
256	512	14000 - 18000	28-36

Table 6.2: Comparison of the ECDLP and the DLP using appropriate embedding degrees

Considering fixed optimal embedding degree, table 6.2 can be reinterpreted as a diagram (figure 6.2) using a logarithmic scale. Note that the gap between the two curves

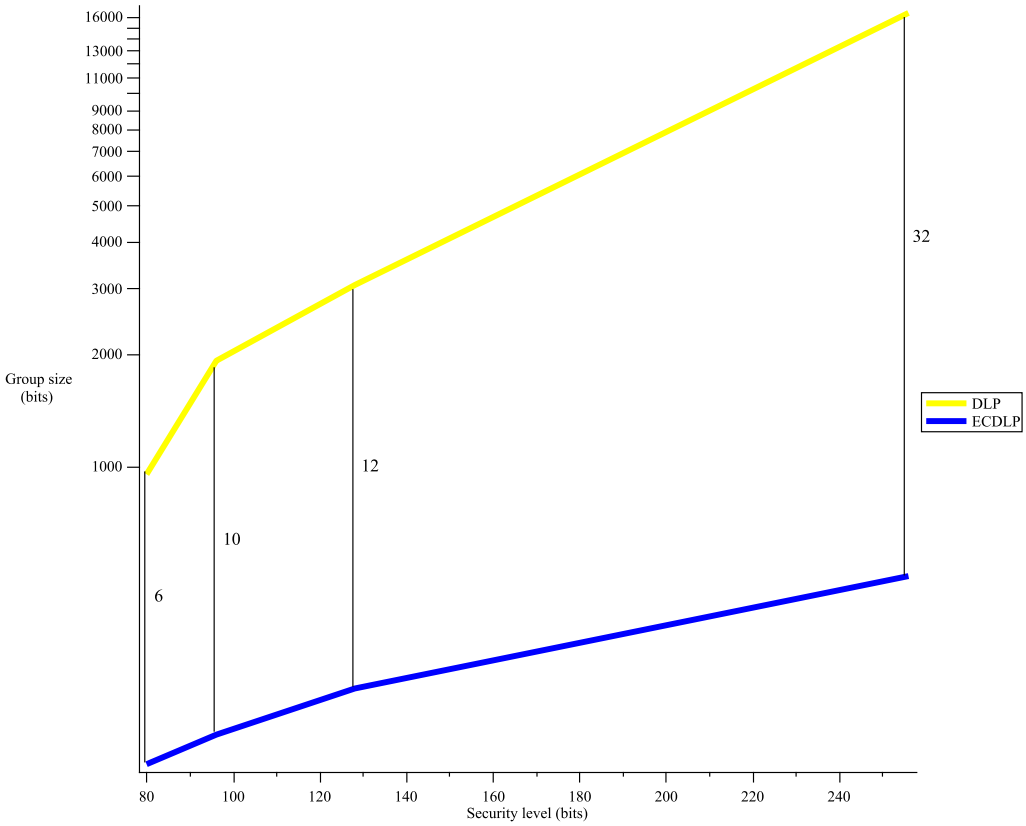

Figure 6.2: Comparison of the ECDLP and the DLP showing optimal embedding degree

widens as the security level and the corresponding embedding degree increase. Since the field into which the pairings map becomes bigger, this may introduce some efficiency issues, especially if no curve with ρ-value close to 1 is found for larger embedding degrees. Regarding the shape of the curve note that the logarithmic scale applies to the ordinate axis only. This explains why the evolution of the group size on the elliptic curve is following a bent curve and not a straight line.

At this stage, we should have grasped the importance of appropriate choices, in order for both the ECDLP and the DLP to be hard, however this may not be enough if one review the results presented in chapter 4 from a new perspective. In fact, this chapter can be reread with security in mind and as such, it presents some major results

on the security of pairings over elliptic curves and more generally over abelian varieties.

Considering Hitt's result [69] not from a mathematical point of view, but from a security perspective, it points out a potential, but crucial, security issue as the minimal embedding field may be a lot smaller than initially thought. In this regard, the new result presented in chapter 4, is of value as it gives some easy conditions to satisfy, in order for the minimal embedding field to be $\mathbb{F}_{q^k}$, ensuring the security of the pairings.

An especially interesting case is when the supersingular curve has embedding degree $k = 4$. In fact, since $\rho \approx 3/2$ is recommended for these curves to achieve an 80bits security level, our result (chapter 4) shows that supersingular $k = 4$ curves are appropriate for this security level for any extension degree m.

We remark that the same consideration also applies to supersingular elliptic curves with embedding degree 6, that is there is no collapse of the minimal embedding field. This implying that Hitt's results does not apply to the curves best fitting the standard 80 bit security level, i.e. the most used in practice.

Once the security of the ECDLP and the DLP have been assessed, the next stage is to clearly express what parameters should be used in order to securely implement pairings. In this regard table 6.3 gives a correspondence between the security level to be achieved and the curve to be used. The first column of the table describes the security level. The second and third columns give the ECDLP group size and the DLP finite field size with corresponding security levels respectively. The fourth and fifth column show the necessary sizes of the fields for corresponding security levels of the DLP in binary and ternary fields respectively. The last column gives the non-supersingular pairing-friendly elliptic curve and finite field with equivalent ECDLP and DLP security.

Analysing more attentively table 6.3 leads to following important remarks:

- It is not possible to balance the security of the ECDLP over $\mathbb{F}_{p^m}$ and the DLP

Security level (in bits)	ECDLP size of r (in bits)	DLP $p > 3$ size of p^k (in bits)	DLP $p = 2$	DLP $p = 3$	Ordinary curves ($\rho \approx 1$)
80	160	960 (6×160)	$\mathbb{F}_{2^{4\times 367}}$	$\mathbb{F}_{3^{6\times 163}}$	$\mathbb{F}_{p^6}$ (MNT)
96	192	1920 (10×192)	$\mathbb{F}_{2^{4\times 613}}$	$\mathbb{F}_{3^{6\times 313}}$	$\mathbb{F}_{p^{10}}$ (Freeman)
128	256	3072 (12×256)	$\mathbb{F}_{2^{4\times 1223}}$	$\mathbb{F}_{3^{6\times 509}}$	$\mathbb{F}_{p^{12}}$ (BN)
192	384	7680 (20×384)	$\mathbb{F}_{2^{4\times 2837}}$	$\mathbb{F}_{3^{6\times 1193}}$	-
256	512	16384 (32×512)	$\mathbb{F}_{2^{4\times 6367}}$	$\mathbb{F}_{3^{6\times 2971}}$	-

Table 6.3: Comparison of the ECDLP and the DLP in finite fields of various characteristic

over $\mathbb{F}_{p^{km}}$, $p = 2, 3$, because of the lower embedding degree and the specialised algorithms for the DLP in fields of low characteristic. For example, over $\mathbb{F}_{2^{163}}$, the optimal embedding degree being 4, one would expect the pairing to map the ECDLP to the DLP in $\mathbb{F}_{2^{652}}$. However, as the current record for computing discrete logarithms in finite fields of characteristic 2 held by Joux is over $\mathbb{F}_{2^{613}}$ [71] and was set some years ago, it is safe to conjecture that with the general advances in computing, the DLP over $\mathbb{F}_{2^{652}}$ is no longer secure. Using supersingular curves, however, does have other advantages and the specific needs for the performance should also be taken into account. This discussion being beyond the scope of this chapter, for more information of this nature, the reader is referred to [57].

- Due to the lower density of prime numbers, it becomes more difficult to find supersingular curves as the security level increases. As shown on figure 6.3, representing the group size as a function of the security level, supersingular curves on characteristic two or three need a much larger group size than ordinary curves on characteristic p to achieve the same security level.

- There is as yet no known construction method for pairing-friendly elliptic curves with $\rho \approx 1$ and $k > 12$. Therefore, the table contains a dash for the 256 and 192 bit security levels as in these cases, the appropriate embedding degrees is required to be larger than 12. However note that the KSS curves with embedding

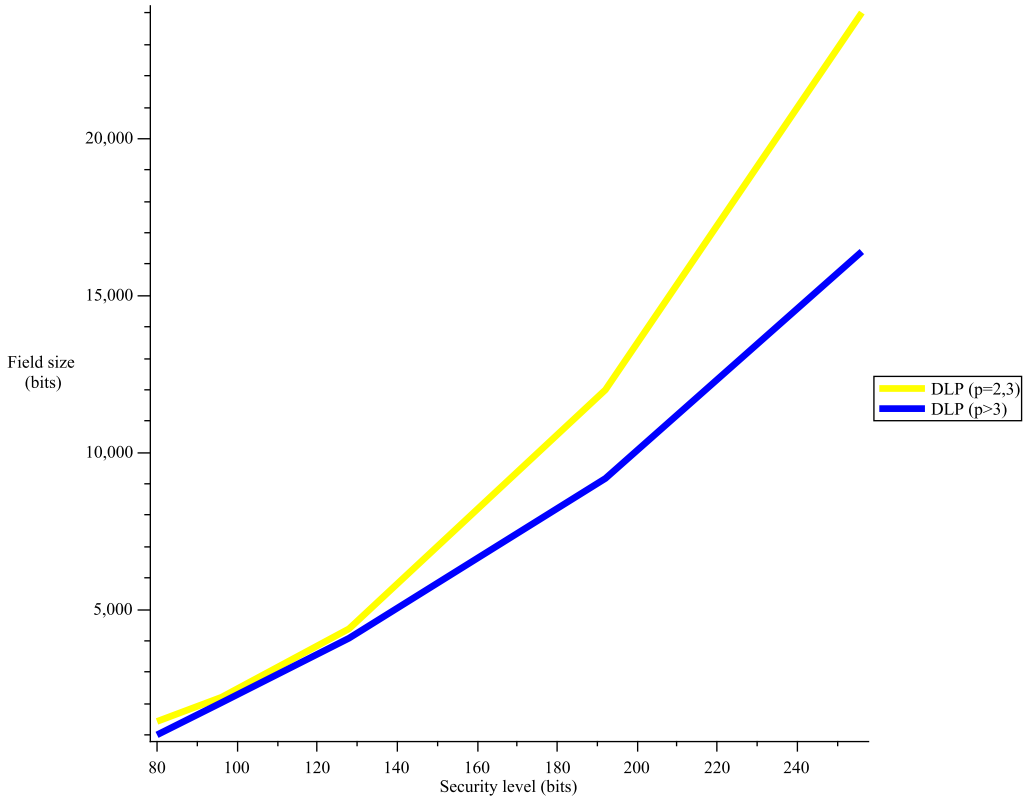

Figure 6.3: Security level of supersingular and ordinary curves

degree $k = 18$ have a ρ-value close to $4/3$, rendering them suitable for the 192 bit security level.

It is interesting to note that, as a pairing maps the ECDLP over $\mathbb{F}_q$ into the DLP over a subgroup of $\mathbb{F}_{q^k}$ of order r, namely the group of rth roots of unity in $\mathbb{F}_{q^k}$, it has been possible to show [59] that it is more efficient to solve the DLP in the cyclotomic subgroup of order $\varphi_k(q)$ of the extension field, rather than in the extension field itself. In fact, the algorithm proposed in [59] is another variation of the index calculus method, which at first glance may seem to improve on the running times of the FFS. However, the cases considered in [59] did not exactly mirror the situation addressed here, as only extension fields having small primes characteristic, larger than 2 or 3, are targeted.

The case of DLP in a subgroup over characteristic 2 or 3 is quickly mentioned, as when m grows, so does the complexity, given by $O((2m)! \cdot q(2^{12m} + 3^{2m}\log(q)) + m^3q^2)$, thus surpassing the complexity of the FFS algorithm. For example, in the case $p = 3$, the complexity is e^a, where $a \approx 57$ and the complexity of the algorithm given in [59] is approximately e^b, where $b \approx 2921$.

One of the major lessons learnt in the examination of the most efficient known algorithms to solve the ECDLP and the DLP, is that those two problems are very hard to solve, the available algorithms having either exponential, or at best, subexponential asymptotic complexity. However as viewed through the implementation of the FFS, a worse asymptotic complexity can result in a more efficient implementation in practice, raising the question of knowing whether it is possible to adapt the FFS [74] to perform better than the NFS, which is burdensome to implement, in the case of ordinary pairing-friendly elliptic curves.

7

Pairings and identity based cryptography

> Why is there Being at all, and not much rather Nothing? That is the question.
>
> M. Heidegger

When using cryptography primitives two main requirements must be met: efficiency and security. As we have seen pairings can be computed very efficiently and are really easy to use in practice as it is possible to automatically generate efficient code [33] to compute them. From a security perspective, if the parameters are properly chosen, both the ECDLP and the DLP are hard to solve, implying the possibility to construct protocols relying on those two similar hard problems. One of the areas benefiting most is Identity Based Cryptography (IBC), first introduced by Shamir [125] in 1984, which aims at simplifying certificate management in conventional public key cryptography.

In IBC, the public key is derived from an identifier, such as an email address or a phone number, while the corresponding private key is created by a private key extraction algorithm which takes the identifier and a master secret as inputs. From 1984 to the beginning of the XXIth century, several Identity Based Encryption (IBE) schemes

were proposed [131, 136, 89], however none of them were fully satisfactory, as most of the solutions proposed were unsafe, requiring the users not to collude. It is only in 2001 that Boneh and Franklin [17], Cocks [25] and Sakai, Ohgishi, and Kasahara [115] presented three revolutionary IBE solutions.

Cocks' scheme is based on the difficulty of distinguishing quadratic residues from non-residues in the ring $\mathbb{Z}_n$ where n is an RSA modulus, and although encryption and decryption are reasonably fast compared to RSA, there exists significant message expansion, which makes it somewhat harder to use in practice. Both Boneh and Franklin (BF-IBE) and Sakai, Ohgishi and Kasahara solutions take advantage of the bilinear property of pairings [108], but the Boneh and Franklin method also has the advantage of defining a well-formulated security model for IBE. As we will see later, their model takes into account the need to be secure against collusion attacks, and as they proved their scheme secure in this new model their IBE scheme received much attention and benefited from some improvements.

One of the main drawback of most such IBE schemes is that, although pairings can be computed efficiently, it still remains slower than traditional public key cryptosystems requiring only multiplications and exponentiations over finite fields. As such, the idea is to lower the number of pairing computations, while keeping the system secure.

Following this idea Callas [21], described a generic framework for constructing an identity-based encryption scheme using a conventional public-key infrastructure. The main idea was to use an identity in order to randomize a key pair generator, such that the Public Key Generator (PKG) could generate conventional key pairs for cryptosystems like RSA [111] or ElGamal [38]. In fact, Callas's scheme is not anymore identity-based in a strict sense, as in his framework, the user can only obtain a public key by accessing the on-line PKG and not by deriving it off-line from a given identifier and some public parameters.

Another idea, due to Tang, Nan and Chen [132], was to combined IBE and some

ElGamal like primitive. Unfortunately, it resulted in an insecure scheme, vulnerable to collusion attacks, as the key generation structure was linear. In this chapter we will present a way to overcome this issue using few pairing computations, but first we start by presenting more formally the cryptographic primitives involved.

7.1 Cryptography

Although coarse, the presentation of cryptology proposed in chapter 1, gives a good idea of what is cryptography about, and what are the main hard problems over which it relies when used with pairings. Therefore we will, in this section, focus only on a few more primitives.

Diffie Hellman Problems: The security of cryptosystems usually relies on hard problems, among which the DLP is probably one of the most well known. The first interesting variant is called the *Computational Diffie-Hellman Problem* (CDH), and consists in computing α^{ab}, given α^a and α^b, for α a generator of a group $\mathbb{G}$.

An other remarkable hard problem, in the Diffie-Hellman class of problems, is given by the *Decisional Diffie-Hellman Problem* (DDH), which states that given a tuples $\langle \alpha, \alpha^a, \alpha^b, \beta \rangle$, one should be able to state whether β is equal to α^{ab} or is a random value.

The last version we consider here is the *Gap Diffie-Hellman Problem* (GDH). The goal is to compute α^{ab}, given α, α^a and α^b, using a DDH oracle returning 0 if a given tuple has a random element and 1 otherwise.

Multivariate Quadratic Problem: First introduced in 1988 by Matsumoto and Imai [88] and later developed by Patarin [106], the MQ problem relies on the difficulty of solving multivariate systems of equations. Let $P_1, \cdots, P_m \in \mathbb{F}_q[x_1, \cdots, x_n]$, be m polynomials of n variables over $\mathbb{F}_q$, such that each of them can be written in the

following form:

$$P_t(x_1, \ldots, x_n) = \sum_{i,j=1}^{n} \beta_{ij}^{(t)} x_i x_j + \sum_{i=1}^{n} \delta_i^{(t)} x_i + \gamma^{(t)}, \text{ with } \beta_{ij}^{(t)}, \delta_i^{(t)}, \gamma^{(t)} \in \mathbb{F}_q$$

Solving a system made of $m \approx n$ such multivariate quadratic equations is proved to be $\mathcal{NP}$-complete even over a field of small characteristic [52, 107].

Symmetric key encryption: When it comes to security it is important to be able to formalise the notion in terms mirroring the real world. For the CCA security, it is done as follows.

Definition 7.1.1. A symmetric key encryption scheme is secure in the $\mathsf{IND\text{-}CCA}$ sense if no probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has a non negligible advantage in the following game:

1. In the setup stage, the challenger randomly chooses a symmetric key sk.
2. In Phase 1, $\mathcal{A}$ starts probing the scheme by querying the encryption oracle $\mathsf{E}(sk, \cdot)$ and the decryption oracle $\mathsf{D}(sk, \cdot)$.
3. In the challenge stage, $\mathcal{A}$ outputs two equal length messages (M_0, M_1) and gets $C = \mathsf{E}(sk, M_\beta)$ for a random bit $\beta \in \{0, 1\}$.
4. In Phase 2, $\mathcal{A}$ issues new queries as in Phase 1 but is disallowed to ask for the decryption of C.
5. In the guess stage, $\mathcal{A}$ eventually outputs a guess β' for β.

$\mathcal{A}$'s advantage is defined by $Adv_{\mathcal{A}}(k) = |\Pr[\beta' = \beta] - 1/2|$.

When the size of an encrypted message is the same as the size of the original message, the scheme is said to be *length preserving*.

Identity Based Encryption: From a structural point of view IBC can be represented by the diagram given in figure 7.1. The PKG has its own public/private key-pair,

that is used to generate the private key of each user of the system. Using some public parameters and a given identity B, A can generate the public key of B, and conversely B can do the same, allowing A and B to start an encrypted communication without the need for any public directory.

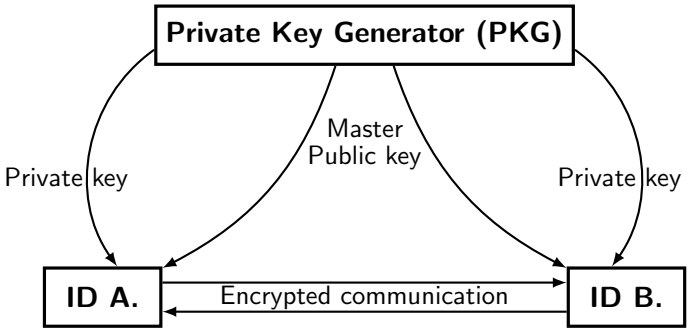

Figure 7.1: Identity based cryptography in practice

This can be formalised in an IBE scheme defined by four algorithms: Setup, Extract, Encrypt, and Decrypt, that can be described as follows:

- *Setup*: takes a security parameter s as input and returns params and $\mathsf{master\text{-}key}$. From a practical viewpoint, params represents the publicly known system parameters, while the $\mathsf{master\text{-}key}$ is only known by the PKG. $\mathcal{M}$ is the message space, and $\mathcal{C}$ the ciphertext space.

- *Extract*: takes as input params, $\mathsf{master\text{-}key}$, and an arbitrary $\mathsf{ID} \in \{0,1\}^*$, to return the associated private key sk.

- *Encrypt*: takes as input params, ID, and $M \in \mathcal{M}$. It returns a ciphertext $C \in \mathcal{C}$.

- *Decrypt*: takes as input params, $C \in \mathcal{C}$, and a private key sk. It returns $M \in \mathcal{M}$ or a reject symbol $\perp$ if C is not a valid ciphertext.

As the users' private key is derived in part from the PKG key-pair, if some users

collude together and are able to recover the PKG's secret key, then the whole system collapses. In the case of an IBE scheme, the IND-CCA security notion cannot fit as it does not involve any security requirement against collusion attack. Therefore, Boneh and Franklin [17] introduced the following definition:

Definition 7.1.2. An IBE scheme is said to be (t, q_E, ϵ)-IND-ID-CCA secure if no t-time adversary making at most q_E private key queries has a non-negligible advantage ϵ in the following game.

1. In the setup stage, the challenger runs the *Setup* algorithm and sends the resulting public parameters to a CCA-adversary $\mathcal{A}$.

2. During phase 1, $\mathcal{A}$ sends queries to two oracles answering as follows:

   - Key extraction oracle: given an extraction query $\langle \mathsf{ID}_i \rangle$, it returns the private key associated to it.

   - Decryption oracle: given a decryption query $\langle \mathsf{ID}_i, C_i \rangle$, it generates the private key d_i associated to ID_i. It then runs algorithm Decrypt to decrypt the ciphertext C_i using d_i. It returns a plaintext $M \in \mathcal{M}$ or a reject symbol $\perp$ indicating an invalid ciphertext.

3. During the challenge stage, $\mathcal{A}$ produces two equal-length messages $M_0, M_1 \in \mathcal{M}$ and a target identity ID on which it wishes to be challenged. The only constraint is that ID did not appear in any private key extraction query in Phase 1. The challenger picks a random bit $\beta \in \{0, 1\}$ and sets $C = \mathsf{Encrypt}(params, M_\beta, \mathsf{ID})$. It sends C as the challenge to the adversary.

4. In Phase 2, $\mathcal{A}$ issues new queries as in Phase 1 but is restricted not to issue a key extraction query on the target identity ID and cannot submit C to the decryption oracle for the identity ID.

5. During the guess stage, $\mathcal{A}$ eventually outputs a bit β' and wins if $\beta' = \beta$.

The advantage of $\mathcal{A}$ against the scheme is given by $Adv_{\mathcal{A}}(k) = |\Pr[\beta' = \beta] - 1/2|$.

We note that the main difference between IND-CCA and IND-ID-CCA relies on the availability of a key extraction oracle in the case of IND-ID-CCA security. This models the possibility that users have to share their secret key during an attack, on a IBE scheme, in the real world.

Random oracle model: The random oracle model was first introduced by Bellare and Rogaway [11], as an idealised security model used to analyse the security of certain cryptographic constructions. It can be seen as a function mapping each input to a random output, in a deterministic way, i.e. if the same input is given twice, then the output will remain the same. Although the security in this model does not necessarily implies security in the real world, it can be used to validate natural cryptographic constructions, or model cryptographic hash functions.

7.2 A new identity based encryption scheme

Using the above primitives we construct a new IBE scheme and discuss its security. But first, we start by explaining the main idea, showing how from ElGamal encryption it is possible to derive an IBE scheme.

7.2.1 Framework

An ElGamal key-pair is defined by $(sk = x, pk = (\mathbb{G}, \alpha, y = \alpha^x))$. Given l such key-pairs $(sk_1, pk_1), \cdots, (sk_l, pk_l)$, we can construct a new key-pair $(SK = \sum_{i=1}^{l} \delta_i x_i, PK = \sum_{i=1}^{l} \delta_i y_i)$, for some δ_i. However, this new secret key is vulnerable to collusion attacks, as the master key can be efficiently determined by solving a linear system of equations [85]. Introducing non-linear terms into the structure would solve this problem, but at the same time it would render the public key computation infeasible in poly-

nomial time as it would imply finding a solution the CDH problem. To overcome this issue, we proceed as follows:

- Pick a pairing-friendly elliptic curve [43] E over $\mathbb{F}_q$. The pairing is defined by the following map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Let P be a generator of $\mathbb{G}_1$ and α a generator of $\mathbb{G}_T$.
- Pick l random numbers $d_1, \ldots, d_l$.
- Compute $U_1 = d_1 P, \ldots, U_l = d_l P$

If we define the secret key as $D = \sum_{i=1}^{l} \delta_i d_i + \sum_{i,j=1}^{l} \beta_{ij} d_i d_j$ for some $\delta_i, \beta_{i,j} \in \mathbb{F}_q$, then we can efficiently compute the corresponding public key using a pairing:

$$\begin{aligned} Q &= \prod_{i=1}^{l} e(U_i, P)^{\delta_i} \prod_{i,j=1}^{l} e(U_i, U_j)^{\beta_{ij}} \\ &= \prod_{i=1}^{l} e(d_i P, P)^{\delta_i} \prod_{i,j=1}^{l} e(d_i P, d_j P)^{\beta_{ij}} \\ &= e(P, P)^{\sum_{i=1}^{l} \delta_i d_i} e(P, P)^{\sum_{i,j=1}^{h} \beta_{ij} d_i d_j} \\ &= \alpha^D \end{aligned}$$

With this setup the secret key is protected by the elliptic curve version of the DLP (ECDLP) [94] and the MQ problem. If an adversary wants to recover the d_i from the U_i he has to solve the ECDLP. If t adversaries collude and get s private keys D_i, they can construct the following system of equations:

$$\begin{cases} \sum_{i=1}^{l} \delta_{1i} d_{1i} + \sum_{i,j=1}^{l} \beta_{1ij} d_{1i} d_{1j} & \equiv D_1 \bmod q \\ \sum_{i=1}^{l} \delta_{2i} d_{2i} + \sum_{i,j=1}^{l} \beta_{2ij} d_{2i} d_{2j} & \equiv D_2 \bmod q \\ \quad \vdots \qquad \ddots \qquad \vdots & \equiv \quad \vdots \\ \sum_{i=1}^{l} \delta_{ti} d_{ti} + \sum_{i,j=1}^{l} \beta_{tij} d_{ti} d_{tj} & \equiv D_t \bmod q \end{cases}$$

As all the δ_i and β_i are chosen randomly this is a system of t multivariate quadratic equations. Solving it would mean solving the MQ problem.

We now describe how to derive a new IBE scheme from these ideas.

7.2.2 A new scheme

The above ideas can be formalised into the four algorithms defining an IBE scheme:
Setup: Given the security parameter s and a parameter $l \in \mathbb{Z}^+$, the algorithm works as follows:

1. Choose the pairings parameters, depending on the security level s to match, and define the pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$. Let $\alpha = e_r(P, P)$ be a generator of $\mathbb{G}_T$.

2. Generate an l-dimensional secret vector $SV = (d_1, \ldots, d_l)$, where d_i is randomly chosen in $\mathbb{F}_q^*$.

3. Generate the corresponding l-dimensional public vector $PV = SV \cdot P = (U_1, \ldots, U_l)$, where $U_i = d_i P$.

4. Choose an $\mathsf{IND\text{-}CCA}$ secure symmetric encryption algorithm SE of key length λ. Encryption is denoted $\mathsf{E}(key, plaintext)$ and decryption $\mathsf{D}(key, ciphertext)$.

5. Let H_0: $\{0,1\}^* \to \{s_1, \ldots, s_t\} \subseteq \{1, \ldots, l\}$ be an identity mapping function, which maps an arbitrary identity string to a t-size subset of $\{1, \ldots, l\}$. Construct a function $H_1 : \{0,1\}^* \to \mathbb{G}_T$ based on H_0, as described below. Choose a cryptographic hash function $H_2 : \mathbb{G}_T \times \mathbb{G}_T \times \mathbb{G}_T \to \{0,1\}^\lambda$.

Thus, the system features the following specifics:

- A master secret key is SV, only known by the PKG.

- Some public parameters $\mathsf{params} = \langle q, \mathbb{G}_1, \mathbb{G}_1, \mathbb{G}_T, P, \alpha, e, PV, H_1, H_2, \mathsf{SE} \rangle$.

- The message space $\mathcal{M} = \{0,1\}^n$ and the ciphertext space $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$.

The H_1 function takes a string as input and relies on the H_0 hash function to generate a subset $\{s_1, \cdots, s_t\} \subset \{1, \cdots, l\}$ of the indexes of PV. Then using these indexes a product of pairings is computed so that $H_1(\mathsf{ID})$ defines the public key.

Extract: For a given identity $\mathsf{ID} \in \{0,1\}^*$, the algorithm works as follows:

1. $H_0(\mathsf{ID}) \to \{s_1, \ldots, s_t\}$, where $s_i \in \{1, \ldots, l\}$.

2. Extract the private key:

$$x_{\mathsf{ID}} = \sum_{1}^{t} d_{s_i} + \sum_{i,j=1}^{t} d_{s_i} d_{s_j}.$$

3. Compute the corresponding public key:

$$y_{\mathsf{ID}} = H_1(\mathsf{ID}) = \prod_{i=1}^{t} e(U_{s_i}, P) \cdot \prod_{i,j=1}^{t} e(U_{s_i}, U_{s_j}).$$

Encrypt: To encrypt $M \in \mathcal{M}$ under the public key ID, do the following:

1. Compute $y_{\mathsf{ID}} = H_1(\mathsf{ID})$.

2. Choose a random r from $\mathbb{F}_q^*$.

3. Compute a symmetric key $sk = H_2(y_{\mathsf{ID}}, \alpha^r, (y_{\mathsf{ID}})^r)$, and set the ciphertext to $C = \langle U, V \rangle = \langle \alpha^r, \mathsf{E}(sk, M) \rangle$.

Decrypt: Upon receiving a ciphertext $C = \langle U, V \rangle \in \mathbb{G}_T \times \{0,1\}^n$, decrypt it using the private key $x_{\mathsf{ID}} \in \mathbb{F}_q^*$ do:

1. Derive the symmetric key $sk = H_2(y_{\mathsf{ID}}, U, U^{x_{\mathsf{ID}}})$.

2. Recover the plaintext $M = \mathsf{D}(sk, V)$.

7.2.3 Considerations on the security of the new scheme

Although a security proof of the new scheme is provided in [24] two major problems occur resulting in an incorrect proof. Indeed the first thing to note is that $H_1(\mathsf{ID}) = \prod_{i=1}^{t} e(U_{s_i}, P) \cdot \prod_{i,j=1}^{t} e(U_{s_i}, U_{s_j})$ cannot be modeled by a random oracle as its output is not completely random. This issue could easy be solved by modeling H_0 instead and use its output to get $H_1(\mathsf{ID})$. However, the second problem is more complex to sort out as the number of queries to the random oracles should be bounded in order to avoid collusion attacks.

In fact, the MQ problem is $\mathcal{NP}$-complete when the number of equations m is roughly equal to the number of unknowns n. However when the number of equations is about $n^2/2$ polynomial time re-linearisation techniques apply [29]. Moreover note that although there has not been any precise theoretical analysis for the cost of solving the MQ problem over a large field it seems that the MQ problem is not easier in this case [13]. Therefore taking $m \approx 0.1n^2$ as suggested in [16] will prevent from polynomial attack.

In the case of our new IBE scheme this means that the number of users handled by the PKG must not exceed $0.1l^2$. For instance if l is taken to be 256 then the new scheme would be safe for a small structure not containing more than 6500 users.

7.3 Discussion

The next stage is to test how efficient this new IBE scheme is in practice. From a theoretical point of view it compares favorably to other IBE schemes proven secure in the random oracle model. Table 7.1 gives a comparison in terms of tightness of the reduction, ciphertext length and, number of operations. The usual notations q_H, q_E, q_D are used for the numbers of hash, extraction and decryption queries, respectively. P denotes a pairing operation, while E denotes an exponentiation in $\mathbb{G}_T$. In order to

achieve a security level of s bits, the group on the elliptic curve must be of size $2s$ and the group resulting from the pairing of size $2ks$, with k the embedding degree of the elliptic curve. The message length is noted n and the tightness of the reduction is given by the complexity of the reduction.

Scheme	Assumption	Reduction	Ciphertext	Encryption	Decryption
BF01 [17]	CBDH	$O(1/(q_E q_H)$	$2s + 2n$	1P+2E	1P+1E
Galindo [51]	CBDH	$O(1/{q_H}^2)$	$2s + n + 80$	1P+2E	1P+ 1E
LQ05 [84]	GBDH	$O(1/q_E)$	$2ks + n$	1P+2E	1P+1E
TightIBE [6]	CBDH	$O(1)$	$2ks + n + 160$	2P+4E	1P+1E
Coron [28]	DSBDH	$O(1)$	$6ks + 2n$	1P+3E	1P+3E
New scheme	-	-	$2ks + n$	2E	1E

- As the original proof of BF-IBE [17] has a flaw, the fixed reduction [114] is mentioned.
- Although, this can be done quite efficiently, LQ05 [84], TightIBE [6] and new scheme, feature an extra computational cost due to the use of an extra symmetric encryption.

Table 7.1: Efficiency comparison between several IBE-schemes

In order to test the efficiency of the scheme from a practical angle, we first need to define more precisely the hash function H_0, which is used to map an arbitrary identity ID to a t-size subset of $\{1, \ldots, l\}$. This can be done as follows:

- Choose t cryptographic hash functions $h_1, \ldots, h_t$.
- Take an arbitrary ID as input, compute $s_i = h_i(\mathsf{ID}) \bmod l$, output $\{s_1, \ldots, s_t\} \subseteq \{1, \ldots, l\}$

As explained previously H_0 maps a string to a subset $\{s_1, \cdots, s_t\}$ of the indexes of the public vector. Then using this subset, a product of pairing involving only the U_{s_i} and P is computed. This defines the H_1 hash function.

The next stage is the choice of the security level. Referring to chapter 6, to achieve an 80 bit security level the group on the elliptic curve should be of size 160. Then, using a supersingular curve with embedding degree 6 allows to match the 80 bit security level on the finite field side. At this stage, being able to rely on an efficient and secure

implementation of pairings [33] is very helpful. It would suffice to check on chapter 6, in order to choose the best fitting curve to reach the expected security level, and then input its parameters and used the efficient implementation returned by the program.

One of the main drawback of the new scheme is the number of pairings that need to be computed during the extraction stage and during the derivation of the public key from the identity. However, as only the product of all the pairings is required, not the computation of each individual pairing, this can be done efficiently, by applying a method proposed by Granger and Smart [58], instead of naively computing each one independently and then multiplying them together. Furthermore note that t cannot be chosen to be too small without compromising the security of the scheme, the MQ problem becoming easier. In our experiments we picked t equal to 24, and l equal to 256. This allows the computation of the pairings to be not too time consuming while preventing attacks targeting sparse systems, i.e. systems such that the probability for a randomly picked variable to appear in an equation is a lot smaller than 1/2, in practice about 1/100 [45].

Table 7.2 give a comparison of BF-IBE and the new scheme. The same IDs and messages were used for the benchmarks, operated on a desktop computer running GNU/Linux with Intel(R) Pentium(R) 4 CPU 3.00GHz processor, 1 GB RAM. In both cases the MIRACL Library [121] was used in order to achieve efficient implementations. We observe that, although the extraction from the new scheme is slower, encryption, and decryption are significantly faster. Once the extraction is done, the key can be stored and then reused later on, leading to a non-negligible gain of time.

Scheme	Extraction	Encryption	Decryption
New Scheme	318.06ms	4.12ms	2.25ms
BF-IBE	30.21ms	34.67ms	31.93ms

Table 7.2: Comparison of our scheme and the BF-IBE

Another important remark concerns the number of pairings used. In fact, Sakai and Kasahara's scheme [23] does not require any pairing computation during the encryption phase, which was a great improvement compared to initial IBE schemes. However, in this chapter we go one step further by not requiring any pairing computation for both encryption and decryption.

8

Pairings and fast hashing

Nothing is more dangerous than an idea, when you only have one.

Alain

Although the new identity based encryption scheme presented in chapter 7 does not require any hash computation into one or both of the two elliptic curve groups involved in the pairing, it is often the case in general [17]. For ordinary curves, the first group, denoted $\mathbb{G}_1$, consists of points on a pairing-friendly elliptic curve E that are defined over the base field $\mathbb{F}_p$, while the second group, denoted $\mathbb{G}_2$, is instantiated as a group of points on a twisted curve E' that have coordinates in some extension field $\mathbb{F}_{p^d}$, where d divides the embedding degree k.

Whereas for the Weil pairing, both input points must have prime order, the Tate pairing and its variants only require one of the input points to be of prime order, as it is sufficient for the other argument to be a coset representative. In fact, the most efficient pairings to date, the ate [67] and R-ate [82] pairings, both variants of the Tate pairing, have the special property of requiring the point of prime order to be in $\mathbb{G}_2$.

Hashing to a point of prime order in $\mathbb{G}_1$ is relatively easy, however, hashing to a prime order point in $\mathbb{G}_2$ requires an additional multiplication by a large cofactor. In this chapter we consider the problem of reducing the cost of hashing to a point of

prime order in $\mathbb{G}_2$. This step may be necessary to ensure efficient implementations of protocols using the Weil, ate or R-ate pairings.

Although points in the group $\mathbb{G}_2$, defined over an extension field may appear cumbersome to handle, Galbraith and Scott [49] observed that arithmetic in $\mathbb{G}_2$ is simpler than it might be thought, as an efficient homomorphism can be exploited. In this chapter we extend their ideas to the related problem of cofactor multiplication in $E'(\mathbb{F}_{p^d})$, which is required to hash an identity to a point of prime order in $\mathbb{G}_2$.

8.1 Twist and number of points

Let E be an elliptic curve defined over a finite field $\mathbb{F}_p$ that has embedding degree $k > 1$, with respect to a prime r, and E' be a twist of E such that r divides $\#E'(\mathbb{F}_{p^d})$ for some $d \mid k$. If $d < k$ we define $\mathbb{G}_2$ to be the unique subgroup of order r on $E'(\mathbb{F}_{p^d})$ [67]. If $d = k$, that is $E \cong E'$, we define $\mathbb{G}_2$ to be the cyclic subgroup of $E[r]$ on which the p-power Frobenius of E acts as multiplication by p.

As the $2 \mid k$ case enables the important denominator elimination optimisation in the pairing calculation [8], we choose k to be even, and as such we can take d, the degree of the extension field, to be $k/2$. Furthermore if the elliptic curve has a CM discriminant of -3 and $6 \mid k$, then we can choose $d = k/6$. Similarly, if the curve has a CM discrimant of -4, and $4 \mid k$, then we can choose $d = k/4$. Clearly the smaller the degree of the extension field $\mathbb{F}_{p^d}$, the easier it will be to manipulate points on $\mathbb{G}_2$.

As recalled in section 3.2, the number of point on an elliptic curve $E/\mathbb{F}_p$ is given by $\#E(\mathbb{F}_p) = p + 1 - t$, with t the trace of the Frobenius and satisfies $\mid t \mid \leq 2\sqrt{p}$. If we now consider points whose coordinates are defined over an extension field $\mathbb{F}_{p^m}$, then

the number of such points on the same elliptic curve [90] is, for instance, given by:

$$\begin{aligned} \#E(\mathbb{F}_{p^2}) &= p^2 + 1 - (t^2 - 2p) \\ \#E(\mathbb{F}_{p^3}) &= p^3 + 1 - (t^3 - 3tp). \end{aligned}$$

In the general case the number of points can be calculated using algorithm 8.1 [90].

Algorithm 8.1 Computation of $\#E(\mathbb{F}_{p^m})$

INPUT: m, p, t: m a positive integer, p a prime, t the trace of Frobenius of an elliptic curve E defined over $\mathbb{F}_p$.
OUTPUT: $\#E(\mathbb{F}_{p^m})$.

1: $\tau_0 \leftarrow 2$
2: $\tau_1 \leftarrow t$
3: **for** $i \leftarrow 1$ **to** $m - 1$ **do**
4: $\quad \tau_{i+1} \leftarrow t \cdot \tau_i - p \cdot \tau_{i-1}$
5: **end for**
6: $q \leftarrow p^m$
7: $\tau \leftarrow \tau_m$
8: **return** $q + 1 - \tau$

A good way to represent the group $\mathbb{G}_2$, is to use an isomorphic group on a twisted curve over the smallest possible extension field. The number of points on the twisted curve can then easily be determined using algorithm 8.1. In the cases of quadratic, quartic and sextic twists, it respectively leads to:

$$\begin{aligned} \text{quadratic: } \#E'(\mathbb{F}_q) &= q + 1 + \tau \\ \text{quartic: } \#E'(\mathbb{F}_q) &= q + 1 - f_1 \text{ where } f_1 = \sqrt{4q - \tau^2} \\ \text{sextic: } \#E'(\mathbb{F}_q) &= q + 1 - (3f_2 + \tau)/2 \text{ where } f_2 = \sqrt{(4q - \tau^2)/3}, \end{aligned}$$

where $q = p^m$ and τ is the trace of the q-power Frobenius on E as calculated in algorithm 8.1. See [67] for more details.

To hash to a point in $\mathbb{G}_2$, the standard idea suggests to first hash to a general point on $E'(\mathbb{F}_{p^d})$ and then multiply by the cofactor $c = \#E'(\mathbb{F}_{p^d})/r$. However, when consid-

ering, for example, a pairing-friendly curve with $k = 10$, $d = 5$ and $r \approx p$, this approach becomes prohibitively slow, as using the quadratic twist implies for the cofactor c, to be of a length, in bits, approximately equivalent to the size of p^4. Therefore, we will now investigate a new way to handle this hashing which will result in a work equivalent to a multiplication by a value less than p, and even in some cases, much less than p.

8.2 Framework

Galbraith and Scott [49, Section 8] have already briefly considered the issue of fast cofactor multiplication of points on $E'(\mathbb{F}_{p^d})$ in the case of BN curves [9]. The idea here, is to generalise and extend their techniques, using the homomorphism $\psi = \phi^{-1}\pi_p\phi$, with $\phi : E' \to E$ the isomorphism which takes us from the twisted curve $E'(\mathbb{F}_{p^d})$ to the isomorphic group on $E(\mathbb{F}_{p^k})$, and π_p the p-power Frobenius map on E. A major remark is to see that $\psi(P)$ can be calculated very quickly.

According to the Galbraith and Scott paper [47, Theorem 1] general points on $E'(\mathbb{F}_{p^d})$ obey the following identity:

$$\psi^2(P) - [t]\psi(P) + [p]P = 0.$$

Applying the usual idea consisting in expressing the cofactor c to the base p we get:

$$c = c_0 + c_1 \cdot p + c_2 \cdot p^2 ...$$

and then using, repeatedly if necessary, the identity

$$[p]P = [t]\psi(P) - \psi^2(P) \tag{8.2.1}$$

it yields the following equation:

$$[c_0 + p(c_1 + p(c_2 + \cdots))]P = [g_0]P + [g_1]\psi(P) + [g_2]\psi^2(P) + \cdots, \quad g_i < p$$

Applying equation 8.2.1 to $[c_1 \cdot p]P$ we get $[c_1 \cdot t]\psi(P) - [c_1]\psi^2(P)$. We note that, t being up to half the size of p (Hasse bound), $c_1 \cdot t$ may be of a size in bits 50% larger than p. Further applications of the homomorphism may therefore be necessary to achieve a complete reduction. Hence, the final result is a recoding of c from a base p representation to a base $\psi(\cdot)$ representation, with all coefficients less than p. The number of terms in the representation increasing with each application of identity 8.2.1, the following identity involving the kth cyclotomic polynomial, Φ_k, may be very useful:

$$\Phi_k(\psi(P)) = 0. \tag{8.2.2}$$

This allows terms of degree greater than or equal to $\varphi(k)$, the Euler totient function, to be replaced with terms of lower degree.

When $k = de$, and $\gcd(d, e) = 1$, the twisting isomorphism ϕ, defining a twist of degree e, can be chosen such that the twisted curve E' is actually defined over $\mathbb{F}_p$, in which case ϕ is defined over $\mathbb{F}_{p^e}$. The cofactor c can then be factored into $h \cdot c_1$, where $c_1 = \#E'(\mathbb{F}_p)$, and the endomorphism $\pi'_p - 1$, where π'_p is the p-power Frobenius map on E', projects into the subgroup of $\#E'(\mathbb{F}_{p^d})$ of order $h \cdot r$. In terms, it means that a point of order r can be obtained at the cost of just one multiplication by h, and as such, the above technique only needs to be applied to the smaller factor, h.

8.3 Fast cofactor multiplication on $\mathbb{G}_2$

Although the basic idea, with minor modifications, can also apply to non-parameterised curves like Cocks-Pinch curves, it benefits from better optimisation when the family of

pairing-friendly elliptic curves can be expressed by three polynomials $t(x)$, $r(x)$ and $p(x)$ representing the prime modulus p, the group order r and the trace t, respectively. Our aim is to exploit this simple form in a systematic way to further speed up the cofactor multiplication required for hashing to $\mathbb{G}_2$. In fact, expressing p as a polynomial $p(x)$, allows both the coefficients and the cofactor c, to be represented and calculated as polynomials in x which in turn leads to further optimisations.

Before proceeding with a few examples we start by formally describing the previous method as an algorithm for reducing the cofactor multiplication to the evaluation of a polynomial of the powers $\psi^i(P)$, with coefficients less than p (algorithm 8.2).

It takes the integer k, and the polynomials $p(x)$, $t(x)$ and $c(x)$, where $p(x)$ and $t(x)$ parameterise the field size of definition and trace respectively of the pairing-friendly curve with embedding degree k. The polynomial $c(x)$ parameterises the hard part of the multiplication to be performed to obtain a point of order r on the twist of the elliptic curve. The first step is to recode $c(x)$ to the base $p(x)$ (lines 3–6) then using this representation of $c(x)$, recode $c(x)$ to the base $\psi(\cdot)$ (lines 8–13). The coefficients of the base $\psi(\cdot)$ representation are computed using the coefficients of the base $p(x)$ representation and the appropriate coefficients of the equation

$$[p^l]P = \sum_{i=0}^{l} \binom{l}{i} t(x)^{l-i}(-1)^i \psi^{l+i}(P),$$

obtained by recursively applying equation (8.2.1). Once $c(x)$ has been written to base $\psi(\cdot)$, the coefficients $g_i(x)$ are checked, such that, if $\deg g_i(x) \geq \deg p(x)$, then the identity $[p]P = [t]\psi(P) - \psi^2(P)$ is reapplied (lines 15–20). Finally the relation (8.2.2) is exploited to obtain a base $\psi(\cdot)$ representation of $c(x)$ of degree $< \varphi(k)$ (lines 22–27).

We now apply this algorithm to certain selected popular families of pairing-friendly elliptic curves, in order to improve the performs of the cofactor multiplication required to hash to a point of order r in $\mathbb{G}_2$.

Algorithm 8.2 Reduction of the cofactor $c(x)$ to base $\psi(\cdot)$

INPUT: $k, p(x), t(x)$, and $c(x)$: embedding degree k and polynomials $p(x)$, $t(x)$, $c(x)$ parameterising the field size, trace, and $\mathbb{G}_2$ cofactor of a pairing-friendly elliptic curve, respectively.

OUTPUT: $g_0(x), g_1(x).....g_{\varphi(k)-1}(x)$: $\deg g_i(x) < \deg p(x)$ will be coefficients of a base $\psi(\cdot)$ representation of the cofactor $c(x)$.

1: $f \leftarrow \lfloor \deg(c(x))/\deg(p(x)) \rfloor$
2: $\diamond$ First express $c(x)$ to the base p
3: **for** $i \leftarrow 0$ **to** f **do**
4: $\quad c_i(x) \leftarrow c(x) \bmod p(x)$
5: $\quad c(x) \leftarrow c(x) \text{ div } p(x)$
6: **end for**
7: $\diamond$ Make first pass to determine the coefficients g_i of $c(x)$ to the base $\psi(\cdot)$, using equation (8.2.1).
8: **for** $j \leftarrow 0$ **to** f **do**
9: $\quad g_{2j} \leftarrow 0$, $g_{2j+1} \leftarrow 0$
10: $\quad$ **for** $i \leftarrow 0$ **to** 1 **do**
11: $\quad\quad g_{j+i} \leftarrow g_{j+i} + \binom{j}{i} t(x)^{j-i}(-1)^i c_j(x)$
12: $\quad$ **end for**
13: **end for**
14: $\diamond$ Make a second pass to finally force all coefficients to have degree $< \deg p$
15: $g_{2f+1} \leftarrow 0$, $g_{2f+2} \leftarrow 0$
16: **for** $j \leftarrow 1$ **to** $2f$ **do**
17: $\quad w(x) \leftarrow g_j(x) \text{ div } p(x)$
18: $\quad g_j(x) \leftarrow g_j(x) \bmod p(x)$
19: $\quad g_{j+1}(x) \leftarrow g_{j+1}(x) + t(x)w(x)$
20: $\quad g_{j+2}(x) \leftarrow g_{j+2}(x) - w(x)$
21: **end for**
22: $\diamond$ Finally exploit equation (8.2.2); a_i is the coefficient of x^i in $\Phi_k(x)$
23: **for** $j \leftarrow 2f+2$ **downto** $\varphi(k)$ **do**
24: $\quad$ **for** $i \leftarrow 1$ **to** $\varphi(k)$ **do**
25: $\quad\quad g_{j-i}(x) \leftarrow g_{j-i}(x) - a_{\varphi(k)-i} \cdot g_j(x)$
26: $\quad$ **end for**
27: $\quad g_j(x) \leftarrow 0$
28: **end for**

8.3.1 MNT curves

MNT pairing-friendly elliptic curves can feature embedding degrees 3, 4 or 6 with ρ value 1. The case of interest for pairing based cryptography, when the ECDLP and the DLP are balanced and secure, is achieved for $k = 6$ (chapter 6). In this case the prime p, the group order r and the trace of Frobenius t parameters are expressed as:

$$\begin{aligned} t(x) &= x + 1 \\ r(x) &= x^2 - x + 1 \\ p(x) &= x^2 + 1. \end{aligned}$$

There exists no x such that the curve generated using these parameters has a CM discriminant of -3, so only a quadratic twist is possible. Here $\mathbb{G}_2$ is a group of points of order r on $E'(\mathbb{F}_{p^3})$. The cofactor is

$$c(x) = \frac{p(x)^3 + 1 + t(x)^3 - 3t(x)p(x)}{r(x)},$$

which in this case works out to be

$$c(x) = x^4 + x^3 + 3x^2.$$

Applying algorithm 8.2 step-by-step we first represent $c(x)$ to the base $p(x)$ (lines 3–6):

$$c(x) = p^2(x) + (x + 1)p(x) + (-x - 2).$$

Now applying equation (8.2.1) to each term involving a power of $p(x)$, and using it to express $[c(x)]P$ in base $\psi(\cdot)$ form gives (lines 8–13):

$$[-x - 2]P + [x^2 + 2x + 1]\psi(P) + [x^2 + x]\psi^2(P) + [-2x - 2]\psi^3(P) + \psi^4(P).$$

One can see that some of the coefficients are still of the same degree as $p(x)$, so one applies equation (8.2.1) again to get (lines 15–20):

$$[-x-2]P + [2x]\psi(P) + [2x]\psi^2(P) + [-x-2]\psi^3(P).$$

All of the polynomial coefficients are now fully reduced modulo $p(x)$. From equation (8.2.2) we know that $\psi^2(P) = \psi(P) - P$, and by substituting this identity twice for $\psi^2(P)$ into the above (lines 22–27), we find that multiplication of a general point P by $c(x)$ can be completed by calculating the point

$$\psi(4xP) - 2xP.$$

This means that the expensive initial multiplication of P by $c(x)$ can be achieved using only one multiplication by x, two point doublings, one application of the homomorphism and a further point addition.

In fact, it can be done even slightly more efficiently. As discussed in section 8.2, since $k = 2 \cdot 3$ and $\gcd(2,3) = 1$, it is possible to choose the quadratic twist E' to be defined over $\mathbb{F}_p$. As such, there must be a subgroup of points of $E'(\mathbb{F}_{p^3})$ which are defined over $\mathbb{F}_p$, that is, the points of $E'(\mathbb{F}_p)$. The number of points on $E'(\mathbb{F}_{p^3})$ must, therefore, have as a factor $p(x)+1+t(x)$, and indeed, in this case $c(x) = (p(x)+1+t(x)) \cdot x^2$. As explained in section 8.2, the first part of the cofactor multiplication by $p(x) + 1 + t(x)$ can be performed by using the Frobenius endomorphism on the twisted curve

$$P \leftarrow \pi'(P) - P,$$

leaving only a further multiplication by x^2. Then, using algorithm 8.2, it is evaluated to simply be $\psi(xP)$.

8.3.2 BN curves

The BN family of pairing-friendly curves [9] has embedding degree 12, and is parameterised as follows:

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

In this case the cofactor multiplication can be done as [49]

$$\psi(6x^2P) + 6x^2P + \psi(P) - \psi^2(P).$$

The major work here is the point multiplication by $6x^2$. As already mentioned, BN curves being plentiful, a usual choice for x is a value having low Hamming weight, which allows to speed up the computation of the pairing using Miller's algorithm. In fact using such a value for x will also speed up the calculation, as the point multiplication will consist largely of point doublings, which are significantly faster than point additions in most curve and point representations.

8.3.3 Freeman Curves

Freeman suggested a construction for pairing-friendly elliptic curves of embedding degree 10 [40].

$$\begin{aligned} t(x) &= 10x^2 + 5x + 3 \\ r(x) &= 25x^4 + 25x^3 + 15x^2 + 5x + 1 \\ p(x) &= 25x^4 + 25x^3 + 25x^2 + 10x + 3. \end{aligned}$$

These curves are much rarer than the BN curves, and unfortunately it is not feasible to choose x to have a particularly small Hamming weight. Furthermore, since the embedding degree is 10, the best that can be done for $\mathbb{G}_2$ is to represent it as a group of points on $E'(\mathbb{F}_{p^5})$. This is a particularly large extension, increasing the chances for the cofactor to be large. In fact $c(x)$, in this case, works out as this polynomial:

$$\begin{aligned} c(x) &= 390625x^{16} + 1562500x^{15} + 4062500x^{14} + 7421875x^{13} + 10750000x^{12} \\ &+ 12593750x^{11} + 12356250x^{10} + 10203125x^{9} + 7178125x^{8} + 4284375x^{7} \\ &+ 2171000x^{6} + 920250x^{5} + 322400x^{4} + 89875x^{3} + 19120x^{2} + 2740x + 217. \end{aligned}$$

Fortunately, it also has $p(x)+1+t(x)$ as a factor, allowing us to apply again the idea in section 8.2. We start by choosing the quadratic twist E' to be defined over $\mathbb{F}_p$, then the multiplication by $p(x)+1+t(x)$ can be handled by the transformation $P \leftarrow \pi'(P) - P$, and so the "hard-part" of the cofactor can be reduced to:

$$\begin{aligned} h(x) &= 15625x^{12} + 46875x^{11} + 93750x^{10} + 128125x^{9} + 138125x^{8} + 116875x^{7} \\ &+ 80875x^{6} + 44875x^{5} + 20225x^{4} + 7075x^{3} + 1880x^{2} + 325x + 31. \end{aligned}$$

Applying our algorithm we find that multiplying P by $h(x)$ can be expressed as:

$$[g_0(x)]P + [g_1(x)]\psi(P) + [g_2(x)]\psi^2(P) + [g_3(x)]\psi^3(P), \tag{8.3.1}$$

where

$$\begin{aligned} g_0(x) &= -5x^2 - 10x - 2; \\ g_1(x) &= -25x^3 - 20x^2 - 10x - 4; \\ g_2(x) &= 3; \\ g_3(x) &= -25x^3 - 10x^2 - 5x. \end{aligned}$$

At this stage we could substitute for x and use a simultaneous multiple point multiplication algorithm [62]. However, equation 8.3.1, together with the g_i, remind us of the problem faced to compute efficiently the final exponentiation in Miller's algorithm (chapter 5), and the solution adopted, which consisted of using Olivos' algorithm, to find the optimal sequence of operations to perform. Therefore, we apply the same strategy here, by calculating xP, $x^2P = x \cdot xP$, $x^3P = x \cdot x^2P$, $\psi^i(P), \psi^i(xP), \psi^i(x^2P)$ and $\psi^i(x^3P)$ for $i = 1$ to 3, and then find the shortest addition sequence including all the coefficients of the g_i. Thus, the calculation first becomes

$$\begin{gathered} [25](-\psi^3(x^3P) - \psi(x^3P)) + [20](-\psi(x^2P)) + [10](-\psi^3(x^2P) - \psi(xP) - xP) \\ +[5](-\psi^3(xP) - x^2P) + [4](-\psi(P)) + [3]\psi^2(P) + [2](-P), \end{gathered}$$

that we consider as

$$25A + 20B + 10C + 5D + 4E + 3F + 2G.$$

While A, B, C, D, E, F and G are calculated using just 4 extra point additions, the optimal way to proceed with the coefficients is to rearrange them such that they form the shortest addition sequence:

$$\{\underline{1}, 2, 3, 4, 5, 10, 20, 25\}.$$

In this case, it is easily done by only adding 1 to the start. Now, we apply Olivos' algorithm [104], [7, Section 9.2] to find the optimal sequence of point additions and doublings to finally proceed to the cofactor multiplication.

$$
\begin{aligned}
T_0 &\leftarrow A + B \\
T_1 &\leftarrow A + D \\
T_0 &\leftarrow 2 \cdot T_0 \\
T_0 &\leftarrow T_0 + C \\
T_0 &\leftarrow 2 \cdot T_0 \\
T_1 &\leftarrow T_0 + T_1 \\
T_0 &\leftarrow T_1 + E \\
T_0 &\leftarrow 2 \cdot T_0 \\
T_0 &\leftarrow T_0 + G \\
T_0 &\leftarrow T_0 + F \\
T_1 &\leftarrow T_1 + F \\
T_0 &\leftarrow 2 \cdot T_0 \\
T_0 &\leftarrow T_0 + T_1.
\end{aligned}
$$

The final result is in T_0. This part of the calculation requires only 9 extra point additions and 4 point doublings.

8.3.4 KSS Curves

Kachisa, Schaeffer and Scott [76] described a new method for generating pairing-friendly elliptic curves.

KSS curves ($k = 8$): The family of $k = 8$ KSS curves is parameterised as follows:

$$\begin{aligned} t(x) &= \frac{2x^3 - 11x + 15}{15} \\ r(x) &= \frac{x^4 - 8x^2 + 25}{450} \\ p(x) &= \frac{x^6 + 2x^5 - 3x^4 + 8x^3 - 15x^2 - 82x + 125}{180}. \end{aligned}$$

For these curves $\rho = 3/2$. As for BN curves, x can be chosen to have a low Hamming weight. Proceeding as above we find

$$\begin{aligned} g_0(x) &= \frac{2x^5 + 4x^4 - x^3 + 50x^2 + 65x - 36}{6} \\ g_1(x) &= \frac{2x^5 + 4x^4 - x^3 - 7x^2 - 25x + 75}{6} \\ g_2(x) &= \frac{-15x^2 - 30x - 75}{6}. \end{aligned}$$

A minor difficulty arises due to the common denominator of 6 which occurs here. However, as noted in chapter 5, in practice r is chosen to be a large prime, implying that $\gcd(6, r) = 1$. Thus, we can complete the hashing to $\mathbb{G}_2$ with the point multiplication $[6 \cdot c(x)]P$, which is also a point of order r. As the denominator can be ignored, we now only need an addition sequence which includes all of the integer coefficients that arise in the numerator of the g_i, in order the complete the calculation:

$$\{1, 2, 4, \underline{5}, \underline{6}, 7, \underline{10}, 15, 25, 30, 36, 50, 65, 75\},$$

Proceeding as for the Freeman curve case, the computation using this addition sequence can be completed with 18 point additions and 5 point doublings.

KSS curves ($k = 18$): The family of $k = 18$ KSS curves can be described by the

following polynomials:

$$\begin{aligned}
t(x) &= \frac{x^4 + 16x + 7}{7} \\
r(x) &= \frac{x^6 + 37x^3 + 343}{343} \\
p(x) &= \frac{x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401}{21}.
\end{aligned}$$

For these curves $\rho = 4/3$ and as for the BN curves x can, in practice, be chosen with a low Hamming weight. Proceeding again as above yields:

$$\begin{aligned}
g_0(x) &= \frac{-5x^7 - 26x^6 - 98x^5 - 381x^4 - 867x^3 - 1911x^2 - 5145x - 5774}{3} \\
g_1(x) &= \frac{-5x^7 - 18x^6 - 38x^4 - 323x^3 - 28x^2 + 784x}{3} \\
g_2(x) &= \frac{-5x^7 - 18x^6 - 38x^4 - 323x^3 + 1029x + 343}{3} \\
g_3(x) &= \frac{-11x^6 - 70x^5 - 98x^4 - 176x^3 - 1218x^2 - 2058x - 686}{3} \\
g_4(x) &= \frac{28x^2 + 245x + 343}{3}.
\end{aligned}$$

Using the same reasoning as in the KSS $k = 8$ case, we evaluate $[3 \cdot c(x)]P$ to remove the denominator of 3. In this case the best addition sequence we could find that includes all of the coefficients was:

$$\{\underline{1}, \underline{2}, \underline{3}, 5, \underline{7}, \underline{8}, 11, 18, 26, 28, \underline{31}, 38, \underline{45}, \underline{69}, 70, \underline{78}, 98, 176, 245, \underline{253}, 323, 343,$$

$$381, \underline{389}, 686, 784, \underline{829}, 867, 1029, 1218, \underline{1658}, 1911, 2058, \underline{4116}, 5145, 5774\}.$$

This allows the completion of the calculation using only 51 point additions and 5 point doublings.

8.4 Discussion

The solution to the problem of multiplying a point P by a large cofactor c mainly relies on the existence of a homomorphism $\psi(\cdot)$ that can be computed efficiently. Thus, it is possible to express c to the base $\psi(P)$, and the initial problem boils down to finding a short addition sequence, as in chapter 5. However, in this case it is slightly more complex as the operations do not occur on a subgroup of $\mathbb{F}_{q^k}$, but on a subgroup of an elliptic curve. As such, the representation of the curve plays an important role: for instance, if doubling or adding a point on $E'(\mathbb{F}_{p^5})$ it is likely that affine coordinates will in fact be faster than any kind of projective coordinates, in which case using the standard short Weierstrass representation, additions may actually be faster than doublings [62]. Hence the remarks from chapter 5, on the use of addition-substraction chains, and on preferring doubling to addition still apply, suggesting that special care must be taken when choosing how to represent the curve. One should then solve the problem of the cofactor multiplication, accordingly to the curve representation initially chosen.

Given an initial hashing to a general point on $E'(\mathbb{F}_{p^d})$, the twist of an ordinary pairing-friendly elliptic curve $E/\mathbb{F}_q$, this new method for deriving a point of prime order r in $\mathbb{G}_2 = E'(\mathbb{F}_{p^d})$, is significantly faster than the naive approach which would require multiplication by a very large cofactor $c = \#E'(\mathbb{F}_{p^d})/r$. As such, it becomes really useful in identity based cryptography where schemes often require the hashing of identities to points on a curve.

9

Conclusion

> A scientific observation always leads to a polemic.
>
> G. Bachelard

Throughout this thesis we have been able to link abstract mathematical results on number theory to applied cryptography. This involved the presentation of pairings as mathematical maps and their study from both efficiency and security perspectives. This allowed the construction of a new IBE scheme taking advantage of pairings, and eventually it was shown that more advanced operations required by their cryptographic use can be achieved efficiently.

Although first used following a destructive strategy, pairings are really helpful in the construction of cryptographic protocols, as one can take advantage of their bilinear property. However, when it comes to implementation, the choice of the parameters is of a major concern in order to ensure both security and efficiency. In fact, since the speed of computations on $J(\mathbb{F}_q)$ is, to an extent, determined by $\#J(\mathbb{F}_q) \approx q^g$ and security is determined by the size of r, for fast implementations one usually wishes to choose J with r as close to $\#J(\mathbb{F}_q)$ as possible, i.e with a ρ-value as close to 1 as possible. In practice one must also take into account the required balance of security for a fixed

k as well as the cost of arithmetic and pairing operations on the elliptic curves under consideration.

The case of supersingular abelian varieties of dimension $g \geq 2$ defined over non-prime fields is interesting, as they have been proposed for use in pairing-based cryptography for being potentially more efficient than supersingular elliptic curves. Furthermore, note that due to index calculus attacks [53, 54] abelian varieties of dimension $g > 4$ are only practical in the context of pairing-based cryptography.

From an application point of view, one of the major implications of the work presented in this thesis is the possibility of automatically generating efficient pairing code, given a single number: the security level. For instance, in the case of a pairing cryptographic protocol, one would only be required to input a security level to be matched and then use the efficient code returned to compute the pairings without any need to understand the hard underlying problems, or any of their related security and efficiency issues. This will surely result in more secure implementations as most of the time security troubles arise from a misunderstanding of the primitives, implying unwanted weaknesses.

Nevertheless, a few areas can still be explored, especially concerning the security of the DLP. As a matter of fact, implementing the NFS seems very challenging, due to the low density of smooth numbers in the sieving space. Thus, as explained previously, a strategy could be to test how the FFS targeting field of medium characteristic, performs in practice.

Another idea could also be to take advantage of the special structures of the field $\mathbb{F}_{q^k}$. Since for the implementation of pairing cryptographic protocols, it is desired that the embedding degree of the curve is of the form $k = 2^i 3^j$ for some small i and j, it would be interesting to study how the composite nature of the extensions can effect the

security.

As suggested by Schirokauer [119], the very specific structure of the polynomials defining the families of pairing-friendly elliptic curves may also allow some variations of the special number field sieve to perform more efficiently in practice than the cumbersome NFS. However, these techniques would probably imply using more sophisticated ideas than the ones presently known.

Appendix A

Pairing-friendly elliptic curves

We give a few examples or pairing-friendly elliptic curves matching the usual security levels.

Supersingular curves:

An example of supersingular elliptic curve over $\mathbb{F}_{2^m}$, where m is an odd integer, with embedding degree $k = 4$ is given by $E : y^2 + y = x^3 + x$. The complexity of the FFS for this curve is given by $L_{2^{4m}}(1/3, (32/9)^{1/3})$. The optimal d and $\mathcal{B}$ values to be used are

$$d = \left\lceil 2\sqrt{\frac{m}{(4/9)^{1/3}(2 + log_2(m))^{2/3}}} \right\rfloor, \text{ and } \mathcal{B} = \left(\frac{16m}{9}\right)^{1/3} (2 + log_2(m))^{2/3}.$$

MNT curves:

An example of an MNT curve with embedding degree 6 is given in [105]. It is defined by

$p = 801819385093403524905014779542892948310645897957,$

$r = 801819385093403524905015674986573529844218487823$

and the elliptic curve has equation $E : y^2 = x^3 - 3x + b$, where

$b = 237567233982590907166836683655522398804119025399$. This curve is suitable for the standard 80 bit security level, also noted AES-80, as $\log p \approx 160$.

Freeman curves:

Taking $p = 61099963271083128746073769567944870354270161646150914794603$ and $r = 61099963271083128746073769567450502219087145916434839626301$ leads to the following Freeman curve $E : y^2 = x^3 - 3x + b$, where
$b = 111277586947145815412995064819820389361361555247649148816 7$. In this case $\log p \approx 196$, which implies matching an AES-98 security level.

BN curves:

BN curves being plentiful it is easy to find an x with low Hamming weight such that p and r are both prime. For example $x = 7530900000000019237$ can be picked. The values obtained are
$p = 115795057838240340066805193894358654649784083976814007840610649712610075073583$
and
$r = 115795057838240340066805193894358654649443797247954006102147570112607854700569$.
This prime p satisfies the congruences $p \equiv 7 \mod 8$, $p \equiv 4 \mod 9$ and $p \equiv 1 \mod 6$, hence by [30], these parameters give a curve $E : y^2 = x^3 + 3$. Both p and r have ≈ 256 bits, implying an AES-128 security level.

Appendix B

Implementations

B.1 A linear sieve

Based on the GMP [2] and NTL [126] libraries, the following simple linear sieve, together with an implementation of the fast Gaussian elimination algorithm, allow to efficiently solve the DLP over prime fields of size p, with $p \approx 70$bits.

```
#include <iostream>
#include <fstream>
#include <vector>
#include <map>

#include <NTL/ZZ.h>
#include <NTL/ZZ_p.h>
#include <NTL/RR.h>

#include "msv.h"

NTL_CLIENT
Miracl precision(2,0);

#define PRECISION 20
#define LOG2(x) log(x)/log(2)
```

```
typedef struct {
vector<int> index;
vector<int> multiplicity;
ZZ rem;
} smooth_dec;


void read_fb(vector<long unsigned> *fctb, string filename);
void sieve_Fp(ZZ p, vector<long unsigned> *fctb, long unsigned clim, long unsigned qlim);
void SmoothTest(smooth_dec *s, ZZ n, vector<long unsigned> *fctb);

int main() {

int i;
ZZ p=to_ZZ("100000000000000000763");

vector<long unsigned> *fctb=new vector<long unsigned>;
read_fb(fctb, "input.txt");

sieve_Fp(p,fctb,800, 2200);

delete fctb;

}


void read_fb(vector<long unsigned> *fctb, string filename) {

/* rem: elements in the factor base are long unsigned int */
long unsigned int tmp;
ifstream in(filename.c_str());

if(!in) cout << "Cannot open file.\n";

while(in >> tmp) fctb->push_back(tmp);

in.close();
```

```
}


void sieve_Fp(ZZ p, vector<long unsigned> *fctb, long unsigned clim, long unsigned qlim) {

modulo((Big)(char*)"10000000000000000763");

long unsigned i, k;

long unsigned c1, c2;
long unsigned q, qpow, nextqpow, nextp;

long unsigned row=0;
long unsigned d1, d2;

double logq;

ZZ prod;
ZZ rel, relinc, n;
ZZ den, denm, num;

std::map<ZZ,int> *extfctb=new std::map<ZZ, int>;
pair<map<ZZ,int>::iterator,bool> ret;

MSM A;

ZZ H=SqrRoot(p)+1;
ZZ J=sqr(H) - p;

/* Get logs of all factor basis primes. */
double *log_fb=new double[sizeof(double)*fctb->size()];
for(i=0; i<fctb->size(); i++) log_fb[i]=LOG2((*fctb)[i]);

/* initialize extended factor base to factor base */
for(k=0;k<fctb->size();k++) extfctb->insert(pair<ZZ,int>(to_ZZ((*fctb)[k]),k));

/* sieving */
for(c1=1;c1<=clim;c1++) {
```

```
double *sieve=new double[sizeof(double)*clim];
if((double)row/(extfctb->size()) >= 1.1)  { delete sieve; break; }

/* denominator and numerator of relations */
den=H+c1;
num=-(J + c1*H);

for(i=0;i<fctb->size();i++) {

q=(*fctb)[i];
logq=log_fb[i];

qpow=q;
while(qpow<=qlim) {

denm=den % qpow;
if(denm == 0) break;

/* InvMod requiers den < qpow */
c2=num*InvMod(denm, to_ZZ(qpow)) % qpow;

if(c2==0) c2=qpow;

nextqpow=qpow*q;

/* Ensure c2 >= c1 to remove redundant relations */
while(c2 < c1)  c2 += qpow;

while(c2 <= clim) {
/* Add logq into sieve for c2 */
sieve[c2] += logq;

/* Test higher powers of q if nextqpow is too large */

if(nextqpow > qlim) {

prod = (J + (c1 + c2)*H + c1*c2) % p;
```

```
nextp = nextqpow;

while(prod % nextp == 0) {
sieve[c2] += logq;
nextp *= q;
}
}

c2 += qpow;

}
qpow = nextqpow;
}
}

rel = den*(H+1);   // the relation
relinc = H+c1;       // add to relation to get next relation

for(c2=1;c2<=clim;c2++) {

n = rel % p;

smooth_dec *s=new smooth_dec;
if(abs(sieve[c2] - floor(LOG2(n))) < 1) { SmoothTest(s, n, fctb); }

if(s->rem==1) {

/* Include each H + c_i in extended factor basis */
ret=extfctb->insert(pair<ZZ,int>(H+c1, extfctb->size()));
if(ret.second==false) d1=ret.first->second;
else d1=extfctb->size()-1;

ret=extfctb->insert(pair<ZZ,int>(H+c2, extfctb->size()));
if(ret.second==false) d2=ret.first->second;
else d2=extfctb->size()-1;

/* Include relation (H + c1)*(H + c2) = fact */
```

```
row++;

for(k=0; k<s->index.size(); k++) A[row][s->index[k]]=s->multiplicity[k];
    if(c1==c2) A[row][d1]=-2;
    else {
A[row][d1]=-1;
A[row][d2]=-1;
}




}

      rel+=relinc;
delete s;

}

if(c1==clim) delete sieve;

}

/* display matrix */
for(i=1;i<=row;i++) {
for (long unsigned j=0; j<extfctb->size(); j++)  cout << A[i](j) <<" ";
cout << endl;
}

delete log_fb;
delete extfctb;

}



void SmoothTest(smooth_dec *s, ZZ a, vector<long unsigned> *fctb) {

long unsigned i, j=0;
```

```
for(i=0;i<fctb->size();i++) {

if(divide(s->rem, a, (*fctb)[i])) {
s->index.push_back(i);
s->multiplicity.push_back(1);
a=s->rem;

/* count multiplicity */
while(divide(s->rem, a, (*fctb)[i])) {
s->multiplicity[j]++;
a=s->rem;
}
j++;
}

}

/* s->rem == 0 <=> no factor */
if(s->rem==0) s->rem=a;

}
```

This code collects the relations and takes the logarithm of the equations resulting in a large sparse matrix of equations that can be solved using structured Gaussian elimination.

B.2 The number field sieve

One of the most time consuming stage in the NFS is the factorisation of the norms of the ideals. Therefore, we efficiently implemented the smooth tests by taking advantage of the GMP [2], GMP-ECM [1] and FLINT [64] libraries.

First, we defined two main structures, allowing to represent the decomposition of a given integer and to set which factorisation method should be used.

```
/* factorization structure */
/* fact -> factors (1st element = the number to be factorized) *
```

```
 * val -> valuation of the corresponding factor *
 * num -> number of factors */
typedef struct nfs_mpz_fact_s {
        mpz_t *fact;
        int *val;
        int num;
} nfs_mpz_fact_t;

/* factorizing methode structure */
/* td, rho, ecm, mpqs = 0 => disable the method */
typedef struct nfs_fact_mth_s {
        int td;
        int prho;
        int ecm;
        int mpqs;
} nfs_fact_mth_t;
```

Then we fixed the number of iterations used when factorising with the Pollard's Rho method to 5000, and set a table containing all the prime integers $2 \leq p \leq 5133$, to be parsed during the trial division process. The factorisation stage is implemented as follows.

```
#include <stdlib.h>
#include <stdio.h>

#include <gmp.h>
#include <ecm.h>
#include <flintlib/mpQS/mpQS.h>
#include <flintlib/F_mpz.h>

#include "factorize.h"

//#define VERBOSE

/* trial division */
void factorize_td(nfs_mpz_fact_t *f, mpz_t n) {

#ifdef VERBOSE
```

```
printf("Trial division: in\n");
#endif

int i;

for(i=0;i<PRIME_LIST_SIZE;i++) {

if(mpz_divisible_ui_p(n,prime_list[i])) {
mpz_init_set_ui(f->fact[f->num],prime_list[i]);
f->val[f->num]=mpz_remove(n,n,f->fact[f->num]);
f->num++;
}

}

#ifdef VERBOSE
printf("Trial division: out\n");
#endif

}


/* definition of f(x) for pollard rho */
void prho_fct(mpz_t x, mpz_t n, int a) {

mpz_mul (x, x, x);
mpz_add_ui (x, x, a);
mpz_mod (x, x, n);

}


/*pollard rho */
void factorize_prho(mpz_t f, mpz_t n) {

#ifdef VERBOSE
printf("Pollar rho: in\n");
gmp_printf("n=%Zd\n",n);
```

```
#endif

int i, a=0;
mpz_t x, y, t1, t2;
mp_limb_t a_limb;

/* define fct(x)=x^2+a (a random) */
while (a == -2 || a == 0) {
mpn_random (&a_limb, (mp_size_t) 1);
a = (int) a_limb;
}

mpz_init_set_si (x, 2);
mpz_init_set_si (y, 2);
mpz_init(t1);
mpz_init_set_si(t2,1);

for(i=0; i<RHO_ITERATIONS && !mpz_cmp_ui(f,1) ;i++) {

prho_fct(x, n, a);
prho_fct(y, n, a); prho_fct(y, n, a);
mpz_sub(t1,x,y);

if(i%50) {
mpz_mul(t2,t2,t1);
mpz_mod(t2,t2,n);
}

else mpz_gcd(f,t2,n);

}

mpz_clear(x);
mpz_clear(y);
mpz_clear(t1);
mpz_clear(t2);

#ifdef VERBOSE
```

```
printf("Pollar rho: out\n");
gmp_printf("n=%Zd, f=%Zd\n",n,f);
#endif

}


/* ecm (ecm.h) */
void factorize_ecm(mpz_t f, mpz_t n, long int B1) {

#ifdef VERBOSE
printf("Ecm: in \n");
gmp_printf("n=%Zd\n",n);
#endif

ecm_params param;
do {
ecm_init(param);
ecm_factor(f,n,B1,param);
B1+=5000;
ecm_clear(param);
}
while(!mpz_cmp_ui(f,1) && B1<=50000);

#ifdef VERBOSE
printf("Ecm: out\n");
gmp_printf("n=%Zd, f=%Zd\n",n,f);
#endif

}


/* pollard p-1 (ecm.h) */
void factorize_pm1(mpz_t f, mpz_t n, long int B) {

ecm_params param;

ecm_init(param);
```

```
param->method=ECM_PM1;

ecm_factor(f,n,B,param);
ecm_clear(param);

}


/* william p+1 (ecm.h) */
void factorize_pp1(mpz_t f, mpz_t n, long int B) {

ecm_params param;

ecm_init(param);
param->method=ECM_PP1;

ecm_factor(f,n,B,param);

ecm_clear(param);

}


/* if n is a perfect power try from 2 to 3 */
int factorize_power(mpz_t f, int *val, mpz_t n) {

int ret=0;
mpz_t rem, t1;

mpz_init(rem);
mpz_init(t1);

while(mpz_perfect_square_p(n)) {
mpz_sqrt(n,n);
*val+=2;
}

while(mpz_perfect_power_p(n)) {
```

```
mpz_rootrem(t1,rem,n,3);
if(!mpz_cmp_ui(rem,0)) {
mpz_set(n,t1);
*val+=3;
}
else {
ret=1;
break;
}
}

mpz_clear(rem);
mpz_clear(t1);

return(ret);

}


/* quadratic sieve (flintlib QS as library) (mpQS.h) */
void factorize_mpqs(F_mpz_factor_t *f, mpz_t n) {

F_mpz_factor_mpQS(f, n);

}


/* add factor to the factorization */
/* pr: 0-> factor to add is not prime, 1-> factor to add is prime */
void factorize_add_factor(nfs_mpz_fact_t *f, mpz_t c, mpz_t n, int pr) {

nfs_fact_mth_t methode={0,1,1,1};

switch (pr) {

case 0:
if(mpz_probab_prime_p(c,10)) {
mpz_init_set(f->fact[f->num],c);
```

```
f->val[f->num]=mpz_remove(n,n,c);
f->num++;
}
else {
mpz_divexact(n,n,c);
factorize(f,c,methode);
}
break;

case 1:
mpz_init_set(f->fact[f->num],n);
f->val[f->num]=1;
f->num++;
break;

}

}



/* main factorization */
/* td: 0-> disable trial division, 1-> enable trial division */
/* 0-> f contains n factorization, 1-> n is prime, 2-> factorization not found */
int factorize(nfs_mpz_fact_t *f, mpz_t n, nfs_fact_mth_t methode) {

int i, ret;
mpz_t c;
mpz_init_set_si(c,1);

/* trial division (only if enabled -> no need during recursion) */
if(methode.td) {
mpz_init_set(f->fact[0],n);
f->val[0]=1;
f->num=1;
if(!mpz_probab_prime_p(n,10)) factorize_td(f,n);
else {
ret=1;
goto CLEAN;
```

```
}
}

/* pollard rho */
if(methode.prho) {
if(mpz_probab_prime_p(n,10)) {
factorize_add_factor(f,c,n,1);
ret=0;
goto CLEAN;
}
else {
do {
mpz_set_si(c,1);
factorize_prho(c,n);
if(mpz_cmp_ui(c,1)) factorize_add_factor(f,c,n,0);
// gmp_printf("rho: c=%Zd n=%Zd\n",c,n);
if(mpz_probab_prime_p(n,10)) {
factorize_add_factor(f,c,n,1);
ret=0;
goto CLEAN;
}
} while(mpz_cmp_ui(c,1));
}
}

// factorize_pp1(factors,tt);
// factorize_pm1(factors,tt);

/* ecm */
if(methode.ecm) {
do {
factorize_ecm(c,n,5000);
if(mpz_cmp_ui(c,1)) factorize_add_factor(f,c,n,0);
if(mpz_probab_prime_p(n,10)) {
factorize_add_factor(f,c,n,1);
ret=0;
goto CLEAN;
}
```

```
if(!mpz_cmp_ui(n,1)) {
ret=0;
goto CLEAN;
}
} while(mpz_cmp_ui(c,1) || mpz_sizeinbase(n,10) <= 28);
}

/* mqfs if size(n)<65digits */
if(methode.mpqs) {
if(mpz_sizeinbase(n,10) <= 65 && mpz_sizeinbase(n,10) > 28) {

F_mpz_factor_t *factors;
factors=malloc(sizeof(F_mpz_factor_t));
factors->fact = malloc(MAX_MPQS_FACTORS*sizeof(mpz_t));
factors->num=0;
for(i=0;i<MAX_MPQS_FACTORS;i++) mpz_init(factors->fact[i]);

factorize_mpqs(factors,n);

for(i=0;i<factors->num;i++) {
mpz_init_set(f->fact[f->num],factors->fact[i]);
f->val[f->num]=mpz_remove(n,n,factors->fact[i]);
f->num++;
mpz_clear(factors->fact[i]);
}

free(factors->fact);
free(factors);

ret=0;
goto CLEAN;

}

/* n is too large */
else {
ret=2;
goto CLEAN;
```

```
}

}

CLEAN:
mpz_clear(c);
return(ret);

}

int smooth(nfs_mpz_fact_t *f, mpz_t n, long signed int B, nfs_fact_mth_t methode) {

int i, ret, big=0;
mpz_t c;
mpz_init_set_si(c,1);

mpz_init_set(f->fact[0],n);
f->val[0]=1;
f->num=1;

/* trial division (only if enabled -> no need during recursion) */
if(methode.td) {
if(!mpz_probab_prime_p(n,10)) factorize_td(f,n);
else {
ret=1;
goto CLEAN;
}
}

/* pollard rho */
if(methode.prho) {
if(mpz_probab_prime_p(n,10) || mpz_cmp_ui(n,1)==0) {
if(mpz_cmp_ui(n,1)) factorize_add_factor(f,c,n,1);
ret=0;
goto CLEAN;
}
else {
do {
```

```
mpz_set_si(c,1);
factorize_prho(c,n);
if(mpz_cmp_ui(c,1)) {
factorize_add_factor(f,c,n,0);
if(mpz_cmp_si(c,B) > 0) big++;
if(big > 1) {
ret=2;
goto CLEAN;
}
// gmp_printf("rho: c=%Zd n=%Zd\n",c,n);
if(mpz_probab_prime_p(n,10) || mpz_cmp_ui(n,1)==0) {
if(mpz_cmp_ui(n,1)) factorize_add_factor(f,c,n,1);
// printf("pollard rho: ");
ret=0;
goto CLEAN;
}
}
} while(mpz_cmp_ui(c,1));
}
}

// factorize_pp1(factors,tt);
// factorize_pm1(factors,tt);

/* ecm */
if(methode.ecm) {
do {
factorize_ecm(c,n,5000);
if(mpz_cmp_ui(c,1)) {
factorize_add_factor(f,c,n,0);
if(mpz_cmp_si(c,B) > 0) big++;
if(big > 1) {
ret=2;
goto CLEAN;
}
if(mpz_probab_prime_p(n,10)) {
factorize_add_factor(f,c,n,1);
ret=0;
```

```
// printf("ecm1: ");
goto CLEAN;
}
if(!mpz_cmp_ui(n,1)) {
ret=0;
// gmp_printf("ecm2: ");
goto CLEAN;
}
}
} while(mpz_cmp_ui(c,1) || mpz_sizeinbase(n,10) <= 28);
}

/* mqfs if size(n)<65digits */
if(methode.mpqs) {
if(mpz_sizeinbase(n,10) <= 65 && mpz_sizeinbase(n,10) > 28) {

F_mpz_factor_t *factors;
factors=malloc(sizeof(F_mpz_factor_t));
factors->fact = malloc(MAX_MPQS_FACTORS*sizeof(mpz_t));
factors->num=0;
for(i=0;i<MAX_MPQS_FACTORS;i++) mpz_init(factors->fact[i]);

factorize_mpqs(factors,n);

for(i=0;i<factors->num;i++) {
mpz_init_set(f->fact[f->num],factors->fact[i]);
f->val[f->num]=mpz_remove(n,n,factors->fact[i]);
f->num++;
mpz_clear(factors->fact[i]);
}

// printf("mpqs: ");
free(factors->fact);
free(factors);

ret=0;
goto CLEAN;
```

```
}

/* n is too large */
else {
ret=2;
goto CLEAN;
}

}

ret=2;

CLEAN:
mpz_clear(c);
return(ret);

}
```

This code allows the decomposition into smooth elements of a given integer. It returns 0 if a complete decomposition is found (even if the integer is not smooth), 1 if the integer is prime and 2 if not all the factors have been found, in particular when more than 1 large factor is found and the remaining factor is composite. For each integer it also generates a structure *nfs_mpz_fact_t* containing the factors, their valuation, and their number. These will then be used to complete the decomposition into prime ideal.

Note that FLINT ships a Multi Precision Quadratic Sieve (MPQS) which is initially compiled as a binary and returns large factors, including duplicates and composites. We adapted this MPQS such that it returns prime factors only, without any duplicates. Nevertheless, in the smoothness test function, MPQS is disabled by default as it targets large composite numbers. All the small factors having been removed using the Pollard's Rho method and the ECM factorisation it means that the factors found by the MPQS will be larger than the smoothness bound, i.e. useless.

References

[1] GMP-ECM Elliptic Curve Method for Integer Factorization. Available from `https://gforge.inria.fr/projects/ecm/`. 85, 135

[2] GNU Multiple Precision Arithmetic Library (GMP). Available from `http://gmplib.org`. 83, 85, 129, 135

[3] MAGMA Computational Algebra System. Website: `http://magma.maths.usyd.edu.au/magma/`. 81, 85

[4] L. Adleman. The function field sieve. In *ANTS I*, volume 877 of *Lecture Notes in Computer Science*, pages 108–121. Springer, 1994. 69

[5] V. Arvind and P. Kurur. On the complexity of computing units in a number field. In *ANTS I*, volume 877 of *Lecture Notes in Computer Science*, pages 72–86. Springer, 2004. 84

[6] N. Attrapadung, J. Furukawa, T. Gomi, G. Hanaoka, H. Imai, and R. Zhang. Efficient identity-based encryption with tight security reduction. In *Cryptology and Network Security*, volume 4301 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2006. 105

[7] R. Avanzi, H. Cohen, D. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman and Hall/CRC, 2006. 54, 120

[8] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryp-

tosystems. In *Advances in Cryptology – Crypto'2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002. 50, 109

[9] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography – SAC'2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2006. 29, 32, 111, 117

[10] P. S. L. M. Barreto, B. Lynn, and M. Scott. On the selection of pairing-friendly groups. In *Selected Areas in Cryptography – SAC'2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer, 2003. 35

[11] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *CCS '93*, pages 62–73. ACM, 1993. 100

[12] N. Benger, M. Charlemagne, and D. Mandell Freeman. On the security of pairing-friendly abelian varieties over non-prime fields. In *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 52–65. Springer, 2009. xiii

[13] L. Bettale, J-C. Faugre, and L. Perret. Hybrid approach for solving multivariate systems over finite fields. 3:177–197, 2010. 104

[14] J-L. Beuchat, N. Brisebarre, J. Detrey, and E. Okamoto. Arithmetic operators for pairing-based cryptography, 2007. Available from `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.73.9782&rep=rep1&type=pdf`. 51

[15] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, volume London Mathematical Society Lecture Note Series. Cambridge University Press, 1999. 28

[16] Bo-Yin Yang, Jiun-Ming Chen. Theoretical analysis of xl over small fields. *ACISP*, 3108:277–288, 2004. 104

[17] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001. 6, 95, 99, 105, 108

[18] J. Bos and M. Coster. Addition chain heuristics. In *Advances in cryptology - CRYPTO '89*, volume 435, pages 400–407. Springer, 1989. 59

[19] F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. *Designs, Codes and Cryptology*, 37:133–141, 2005. 30

[20] J. Buchman. A subexponential algorithm for the determination of class groups and regulators of algebraic number fields. In *Séminaire de Thorie des Nombres*, pages 27–41, 1990. 84

[21] J. Callas. Identity-based encryption with conventional public-key infrastructure. *In 4th Annual PKI Research and Develop Workshop, number 7224 in Interagency Reports*, pages 102–115, 2005. 95

[22] G. Cardona and E. Nart. Zeta function and cryptographic exponent of supersingular curves of genus 2. In *Pairing-Based Cryptography — Pairing 2007*, volume 4575 of *Springer Lecture Notes in Computer Science*, pages 132–151, 2007. 46

[23] L. Chen and Z. Cheng. Security proof of sakai-kasahara's identity-based encryption scheme. In *Proceedings of Cryptography and Coding 2005*, volume 3706 of *Lecture Notes in Computer Science*, pages 442–459, 2005. 107

[24] Y. Chen, M. Charlemagne, Z. Guan, J. Hu, and Z. Chen. Identity-based encryption based on DHIES. In *ASIACCS*, pages 82–88. ACM, 2010. xiii, 104

[25] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363. Springer, 2001. 95

[26] H. Cohen. *A Course in Computational Algebraic Number Theory*, volume 138 of *Graduate Texts in Mathematics*. Springer, 1993. 77

[27] D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory*, 30:587–594, 1984. 81

[28] J-S. Coron. A variant of boneh-franklin IBE with a tight reduction in the random oracle model. *Designes, Codes and Cryptography*, 50:115–133, 2009. 105

[29] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in cryptology – EUROCRYPT '2000*, volume 1807, pages 392–407, 2000. 104

[30] A. Devegili, M. Scott, and R. Dahab. Implementing cryptographic pairings over Barreto-Naehrig curves. In *Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 197–2007. Springer, 2007. 128

[31] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, November 1976. Also available from `http://www.cs.purdue.edu/homes/ninghui/courses/Fall04/lectures/diffie-hellman.pdf`. 3

[32] C. Doche and T. Lange. Arithmetic of elliptic curves. In *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, pages 267–302. Chapman & Hall/CRC, Boca Raton, FL, 2006. 34

[33] L. Dominguez and M. Scott. Automatic generation of optimised cryptographic pairing functions. In *SPEED-CC 2009*, 2009. Also available from `http://www.hyperelliptic.org/SPEED/record09.pdf`. 60, 94, 106

[34] L. Downey and Sethi. Computing sequences with addition chains. *Siam Journal of Computing*, 3:638–696, 1981. 59

[35] S. Duquesne and G. Frey. Background on pairings. In *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, pages 115–124. Chapman & Hall/CRC, Boca Raton, FL, 2006. 37

[36] H. Edwards. Dedekind's invention of ideals. *Journal of the London Mathematical Society*, 15:8–17, 1983. 9

[37] D. Eisenbud and J. Harris. *The geometry of schemes*, volume 197 of *Graduate Texts in Mathematics*. Springer, 1999. 11

[38] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO '84 on Advances in cryptology*, pages 10–18. Springer, 1985. 95

[39] F. Vercauteren F. Hess, N. Smart. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10), October 2006. Also available from `http://eprint.iacr.org/2006/110.pdf`. 35

[40] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. In *ANTS VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 452–465. Springer, 2006. 117

[41] D. Freeman. Constructing pairing-friendly elliptic curves with embedding degree 10. *Algebraic Number Theory Symposium ANTS-VII*, Lecture Notes in Computer Science(4575):152–176, 2007. 29

[42] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing friendly elliptic curves. *Cryptology ePrint Archive, Report 2006/372*, 2006. http://eprint.iacr.org. 26

[43] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, 23(2):224–280, 2010. Also available from `http://eprint.iacr.org/2006/372`. 27, 31, 32, 39, 88, 101

[44] W. Fulton. *Algebraic curves, An Introduction to Algebraic Geometry.* Available from `http://www.math.lsa.umich.edu/~wfulton/CurveBook.pdf`. 13, 19

[45] C. Jefferson G. Bard, N. Courtois. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials, 2007. Availbale from `http://eprint.iacr.org/2007/024`. 106

[46] S. Galbraith. Advances in elliptic curve cryptography - Pairings. 2005. 23, 27, 34

[47] S. Galbraith, X. Lin, and M. Scott. Endomorphisms for faster elliptic curve cryptography on a large class of curves. In *Advances in Cryptology - EUROCRYPT '09*, Lecture Notes in Computer Science, pages 518–535. Springer, 2009. 111

[48] S. Galbraith, J. McKee, and P. Valencia. Ordinary abelian varieties having small embedding degree. *Finite Fields and their Applications*, 13:800–814, 2007. Also availbale from `http://eprint.iacr.org/2004/365.pdf`. 49

[49] S. Galbraith and M. Scott. Exponentiation in pairing-friendly groups using homomorphisms. In *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 211–224. Springer, 2008. 109, 111, 117

[50] S. Galbraith and N. Smart. A cryptographic application of Weil descent. In *Proceedings of the 7th IMA International Conference on Cryptography and Coding*, pages 191–200, London, UK, 1999. Springer. 5

[51] D. Galindo. Boneh-franklin identity based encryption revisited. In *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming*, volume 3580 of *Lecture Notes in Computer Science*, pages 791–802. Springer, 2005. 105

[52] M. Garey and D. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979. 97

[53] P. Gaudry. Index calculus for abelian varieties and the elliptic curve discrete logarithm problem. To appear in *J. Symbolic Computation*. Also available from `http://eprint.iacr.org/2004/073`. 125

[54] P. Gaudry, F. Hess, and N. P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. *J. Cryptology*, 15(1):19–46, 2002. 125

[55] R. Granger, A. Holt, D. Page, N. Smart, and F. Vercauteren. Function field sieve in characteristic three. In *ANTS V*, volume 3076 of *Lecture Notes in Computer Science*, pages 223–234. Springer, 2004. 72

[56] R. Granger, D. Page, and N. Smart. High security pairing-based cryptography revisited. In *ANTS VII*, volume 4076 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 2006. 51

[57] R. Granger, D. Page, and M. Stam. On small characteristic algebraic tori in pairing-based cryptography. *LMS Journal of Computation and Mathematics*, 9:64–85, March 2006. Also available from `http://eprint.iacr.org/2004/132.pdf`. 91

[58] R. Granger and N. Smart. On computing products of pairings. Available from `http://eprint.iacr.org/2006/172`. 106

[59] R. Granger and F. Vercauteren. On the discrete logarithm problem on algebraic tori. In *Advances in Cryptology – CRYPTO 2005*, volume Lecture Notes in Computer Science, pages 66–85. Springer, 2005. 92, 93

[60] L. Gruson. Surfaces de riemann. Cours de master 1, UVSQ. 21

[61] D. Hankerson, A. Menezes, and M. Scott. Software implementation of pairings. CACR Technical Report, 2008. `http://www.cacr.math.uwaterloo.ca/`. 51

[62] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curves Cryptography*. Springer, 2004. 119, 123

[63] G. Hardy. *A mathematician's apology*. Cambridge: University Press, 1940. 1

[64] W. Hart. FLINT: Fast Library for Number Theory. Available from `http://www.flintlib.org/`. 85, 135

[65] T. Hayashi, N.Shinohara, L.Wang, S. Matsuo, M.Shirase, and T.Takagi. Solving a 676-bit discrete logarithm problem in $\mathbb{F}_{3^{6n}}$. In *PKC 2010*, Lecture Notes in Computer Science, pages 351–367. Springer, 2010. Also available from `http://eprint.iacr.org/2010/090.pdf`. 82

[66] L. Hei, J. Dong, and D. Pei. Implementation of cryptosystems based on Tate pairing. *Journal of Computer Science & Technolgy*, 20(2):264–269, 2005. 34, 51

[67] F. Hess, N. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10):4595–4602, 2006. 108, 109, 110

[68] M. Hindry. Introduction to abelian varieties and Mordell-Lang conjecture, 1997. Available from `http://www.math.jussieu.fr/~hindry/abvarmodel.pdf`. 21

[69] L. Hitt. On the minimal embedding field. In *Pairing-Based Cryptography — Pairing 2007*, volume 4575 of *Springer Lecture Notes in Computer Science*, pages 294–301. Springer, 2007. 36, 37, 49, 90

[70] L. Hitt O'Connor, G. McGuire, M. Naehrig, and M. Streng. CM construction of genus 2 curves with p-rank 1. Available from `http://eprint.iacr.org/2008/491`. 49

[71] A. Joux. Discrete logarithms in $\mathbb{F}_{2^{607}}$ and $\mathbb{F}_{2^{613}}$. From the number theory list archives (September 2005) `http://listserv.nodak.edu/archives/nmbrthry.html`. 83, 91

[72] A. Joux. A one round protocol for tripartite Diffie-Hellman. *Journal of Cryptology*, 17(4):263–276, 2004. 6

[73] A. Joux and R. Lercier. The function field sieve is quite special. In *ANTS V*, Lecture Notes in Computer Science, pages 431–445. Springer, 2002. 69, 70, 73

[74] A. Joux and R. Lercier. The function field sieve in the medium prime case. In *Adavances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 254–270. Springer, 2006. 82, 93

[75] A. Joux, R. Lercier, N. Smart, and F. Vercauteren. The number field sieve in the medium prime case. In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 323–341, 2006. 74, 75, 79, 80, 81, 83, 86

[76] E. Kachisa, E. Schaefer, and M. Scott. Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 126–135. Springer, 2008. Also available from `http://eprint.iacr.org`. 30, 31, 120

[77] Neal Koblitz. Elliptic curves cryptosystems. *Math. Comp.*, 48(5):203–209, 1987. 4

[78] V. Kreinovich and L. Longpré. How important is theory for practical problems? A partial explanation of Hartmanis' observation. *Bulletin of the EATCS*, 71:160–164, 2000. 1

[79] B. LaMacchia and A. Odlyzko. Solving large sparse linear systems over finite fields. volume 537 of *Lecture Notes in Computer Science*, pages 109–133. Springer, 1991. 67

[80] C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau Standards*, 49:33–53, 1952. 67

[81] S. Lang. *Algebra*, volume 211 of *Graduate Texts in Mathematics*. Springer, New York, revised third edition, 2002. 39

[82] E. Lee, H-S. Lee, and C-M. Park. Efficient and generalized pairing computation on abelian varieties. *IEEE Transactions on Information Theory*, 55:1793–1803, 2009. Also available from `http://eprint.iacr.org/2008/040`. 35, 108

[83] A. Lenstra. Unbelievable security: Matching AES security using public key systems. In *Proceedings Asiacrypt 2001, LNCS 2248, Springer-Verlag 2001, 6786*, volume 2248 of *Lecture Notes in Computer Science*, pages 67–86. Springer, 2001. 88

[84] B. Libert and J-J. Quisquater. Identity based encryption without redundancy. In *Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 285–300. Springer, 2005. 105

[85] T. Lickteig. Gaussian elimination is optimal for solving linear equations in dimension two. *Information Processing Letters*, 22(6):277–279, 1986. 100

[86] F. Luca, A. México, and I. Shparlinski. Elliptic curves with low embedding degree. *Journal of Cryptology*, 19:553–562, October 2006. 27

[87] R. Matsumoto. Using C_{ab} curves in the function field sieve. *IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E82-A:551–552, 1999. 70

[88] T. Matsumoto and H. Imai. Public quadratic polynomial-tuples for efficient signature-verification and message-encryption. In *Advances in Cryptology – EUROCRYPT' 88*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer, 1988. 96

[89] U. Maurer and Y. Yacobi. Non-interactive public-key cryptography. In *Advances in Cryptology – EUROCRYPT '91*, volume 1751 of *Lectures in Computer Science*, pages 498–507. Springer, 1991. 95

[90] A. Menezes. *Elliptic Curve Public Key Cryptosystems.* Kluwer Academic Publishers, 1993. 110

[91] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39:1639–1646, 1993. 5, 27, 61, 62

[92] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography.* CRC Press, 1996. 51

[93] V. Miller. Short programs for functions on curves, 1986. 33

[94] V. Miller. Use of elliptic curves in cryptography. In *Advances in cryptology – CRYPTO '85*, volume 263 of *Lecture notes in computer sciences*, pages 417–426. Springer, 1986. 4, 101

[95] J. Milne. *Abelian varieties.* Course notes, available from `http://www.jmilne.org/math/CourseNotes/AV.pdf`. 15, 21, 22

[96] J. Milne. Abelian varieties. In *Arithmetic Geometry*, pages 103–150. Springer, 1986. 26, 37

[97] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E84-A:1234–1243, 2001. 28

[98] F. Morain. Building cyclic elliptic curves modulo large primes. In *Advances in cryptology – EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 328–336. Springer, 1991. 27

[99] M. Naehrig, P. Barreto, and P. Schwabe. On compressible pairings and their computation. In *Progress in Cryptology - AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2008. 51

[100] J. Neukrich. *Algebraic Number Theory*, volume 322 of *Grundlehren der mathematischen Wissenschaften.* Springer, 1999. 11, 15, 76

[101] NIST. Computer security. Available from `http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf`. 4, 88

[102] Y. Nogami, M. Akane, Y. Sakemi, H. Kato, and Y. Morikawa. Integer variable X-based ate pairing. In *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 178–191. Springer, 2008. 55

[103] J. Oesterlé. Introduction à la théorie des nombres. Cours de master 2, Paris VI. 10, 11, 12

[104] J. Olivos. On vectorial addition chains. *Journal of Algorithms*, 2:13–21, 1981. 54, 120

[105] D. Page, N. Smart, and F. Vercauteren. A comparison of MNT curves and supersingular curves. *Applicable Algebra in Engineering, Communication and Computing*, 17:379–392, 2006. Also available from `http://eprint.iacr.org/2004/165.pdf`. 127

[106] J. Patarin. Hidden field equations and isomorphisms of polynomials: two new families of asymmetric algorithms. In *Advances in Cryptology – EUROCRYPT' 96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996. 96

[107] J. Patarin and L. Goubin. Trapdoor one-way permutations and multivariate polynominals. In *ICICS '97: Proceedings of the First International Conference on Information and Communication Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 356–368. Springer, 1997. 97

[108] K. Paterson. Advances in elliptic curve cryptography - Cryptography from pairings. 2005. 95

[109] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $\mathbb{F}_p$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, January 1978. 64

[110] J. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32:918–924, 1978. 62

[111] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. 95

[112] K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology. In *Advances in Cryptology — CRYPTO 2002*, volume 2442 of *Springer Lecture Notes in Computer Science*, pages 336–353, 2002. 36, 37, 38, 48

[113] K. Rubin and A. Silverberg. Using abelian varieties to improve pairing-based cryptography. To appear in *Journal of Cryptology*, 2009. 36, 38

[114] Z. Rui and H. Imai. Improvements on security proofs of some identity based encryption schemes. In *Information Security and Cryptology*, volume 3822 of *Lecture Notes in Computer Science*, pages 28–41, 2005. 105

[115] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing over elliptic curves. *The 2001 Symposium on Cryptography and Information Security, Japan*, 45:26–28, 2001. 95

[116] P. Samuel. *Théorie algébrique des nombres*. Hermann, 1971. 9

[117] T. Satoh and K. Araki. fermat quotients and the poynomial time discrete logarithm for anomalous elliptic curves, 1997. 5

[118] O. Schirokauer. Discrete logarithms and local units. *Philosophical Transactions: Physical Sciences and Engineering*, 345:409–423, 1993. 80

[119] O. Schirokauer. Using number fields to compute logarithms in finite fields. *Mathematics of Computation*, 231:1267–1283, 2000. 74, 126

[120] M. Scott. Implementing cryptographic pairings. Available from `ftp://ftp.computing.dcu.ie/pub/resources/crypto/pairings.pdf`. 34

[121] M. Scott. Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL). Available from `http://www.shamus.ie/`. 83, 106

[122] M. Scott and P. Barreto. Compressed pairings. In *Advances in Cryptology – Crypto' 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 140–156. Springer, 2004. Also available from `http://eprint.iacr.org/2004/032/`. 32, 51

[123] M. Scott, N. Benger, M. Charlemagne, L. Dominguez Perez, and E. Kachisa. Fast hashing to $\mathbb{G}_2$ on pairing-friendly curves. In *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2009. xiii

[124] M. Scott, N. Benger, M. Charlemagne, L. Dominguez Perez, and E. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 78–88. Springer, 2009. xiii

[125] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in cryptology – CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984. 94

[126] V. Shoup. NTL: A Library for doing Number Theory. Available from `http://www.shoup.net/ntl/`. 83, 129

[127] J. Silverman. *The Arithmetic of Elliptic Curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, 1986. 13, 16, 18, 27, 32, 37

[128] N. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12:193–196, 1999. 5

[129] M. Stam and A. Lenstra. Efficient subgroup exponentiation in quadratic and sixth degree extensions. In *CHES 2002*, volume 2523 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2002. 51

[130] A. Sutherland. Computing Hilbert class polynomials with the Chinese remainder theorem, 2009. Available from `http://arxiv.org/pdf/0903.2785v3`. 28

[131] H. Tanaka. A realization scheme for the identity-based cryptosystem. In *Advances in cryptology – CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 340–349. Springer, 1988. 95

[132] W. Tang, X. Nan, and Z. Chen. Combined public key cryptosystem. In *Proceedings of IEEE 12th International Conference on Software, Telecommunications and Computer Networks – SoftCOM '04*, 2004. 95

[133] E. Teske. Speeding up pollard's rho method for computing discrete logarithms. In *ANTS III*, Lecture Notes in Computer Science, pages 541–554. Springer, 1998. 81, 87

[134] C. Thiel. Under the assumption of the Generalized Riemann Hypothesis verifying the class number belongs to $\mathcal{NP} \bigcap co - \mathcal{NP}$. In *ANTS I*, volume 877 of *Lecture Notes in Computer Science*, pages 234–247. Springer, 1994. 84

[135] E. Thome. Computation of discrete logorithms in $\mathbb{F}_{2^{607}}$. From the number theory list archives (February 2002) `http://listserv.nodak.edu/archives/nmbrthry.html`. 81, 85

[136] S. Tsujii, T. Itoh, and K. Kurosawa. ID-based cryptosystem using discrete logarithm problem. *Electronics Letters*, 23:1318–1320, 1987. 95

[137] P. van Oorschot and M. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12:1–28, 1999. 81, 87

[138] L. Washington. *Introduction to Cyclotomic Fields*, volume 83 of *Graduate Texts in Mathematics*. Springer, 1997. 24, 25

[139] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32:54–62, 1986. 67